NASA Contractor Report 189739

H-P ADAPTIVE METHODS FOR FINITE ELEMENT ANALYSIS OF AEROTHERMAL LOADS IN HIGH-SPEED FLOWS

H.J. Chang, J.M. Bass, W. Tworzydlo, and J.T. Oden

COMPUTATIONAL MECHANICS COMPANY, INC.
Austin, Texas

Contract NAS1-18746
January 1993

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

Contents

Project Summary

Over the past four years the Computational Mechanics Company, Inc. has been the principal investigator into the development of a new class of computational methods for modeling hypersonic viscous flows in two- and three-dimensional domains. The ultimate goal of this project is to provide NASA with a highly accurate computational tool for resolving fine scale flow features typical of shock wave interactions and strong viscous interaction regions in hypersonic flows. Toward this end, a research and development effort was put forth in the area of adaptive computational finite element methods for high speed flows based on unstructured mesh concepts and employing local estimates of the solution error to optimally change the computational grid to minimize the numerical error. The approach developed here, which combines h-adaptive and anisotropic p-enrichment mesh modification procedures, has proven for certain classes of problems to provide exponential rates of convergence of the solution using possibly an order of magnitude fewer degrees of freedom than conventional methods while obtaining the same level of computational accuracy.

During the first phase of this project (years 1–3), work was focused on a number of research topics which were crucial to the success of h-p adaptive finite element methods for hypersonic flows. Many of these issues were resolved and the results are presented again in detail in the body of this report for completeness. Summarizing some of the more significant developments of this first phase of the effort:

- The development of the first h-p finite element data structure for quadrilateral and hexahedral elements.
- The development of an h-p adaptive package that includes
  - A local error estimation capability for driving the adaptive strategy.
  - A spectral enrichment and h-refinement methodology to change the structure of the computational grid.
- The formulation of a generalized methodology for handling nodal point constraints for higher order polynomials. (Note that this difficulty does not occur with either h-refinement or p-enrichment individually.)
- The development of an algorithm for manual anisotropic enrichment of both two-dimensional and three-dimensional elements.
- The formulation and implementation of a version of a preconditioned block Jacobi-GMRES method for higher order spectral elements.

- The formulation and implementation of a one-step Taylor-Galerkin solution algorithm for h-p methods.
- The formulation and implementation of an implicit two-step Taylor-Galerkin solution algorithm which solves first an Euler step followed by a viscous step.
- An investigation of artificial dissipation mechanisms appropriate for h-p adaptive computational methods with possibly highly distorted elements or elements with high aspect ratios.

In addition to these efforts on research-oriented topics, a user friendly, graphics oriented, interface for the two- and three-dimensional codes was developed. This interface includes both a batch and interactive option and full graphics capabilities for displaying the solutions, extracted quantities, and plots of the local estimates of the computational error. A simple grid file interface was also developed to read in neutral grid files generated either by the GAMMA2D or GAMMA3D codes, developed in-house at COMCO, or by PATRAN. (The formats necessary for PATRAN and the neutral file interface are discussed in the grid generation section of the user manual.)

The second phase of this effort, conducted over the last 12 months, has focused on two special research and development topics which are in general related to the performance of the flow solver. These topics include:

- The development of implicit/explicit computational methods (in two dimensions) for integrating the Navier Stokes equations forward in time.
- The development of computational methodologies and algorithms which provide for automated directional p-enrichment of the computational mesh.

A third topic on which we dedicated considerable computer resources was a continued investigation of artificial dissipation mechanism for h-p adaptive computational methods. In particular, numerous test cases were run for the Mach 14 Holden problem using various artificial mechanism to determine an appropriate model for capturing the recirculation bubble and other fine features of the solution.

All of the capabilities and options developed during the first and second phases of the effort have been implemented and/or tested in a two-dimensional and/or a three-dimensional finite element code. Using these codes a number of test cases have been solved to verify the functionality of the software. These benchmark cases include flow past a blunt body with an incident shock to produce a Type IV (Edney) interaction, a Holden problem with inflow Mach number 14, and a rearward facing step with a strong expansion. Experimental data

on each of these benchmark problems is available and comparisons with this data are made throughout the results section of this report.

1 Introduction

The commitment to develop the National Aero-Space Plane and Maneuvering Reentry Vehicles has generated resurgent interest in the technology required to design structures for hypersonic flight. As these vehicles cruise, accelerate and/or decelerate in the atmosphere, highly complex patterns of shock wave interactions and shock wave boundary layer interactions develop which produce severe local pressures and extreme local heating rates. To provide adequate safety factors in the thermal-structural designs an accurate determination of the aerothermal loads, especially in the local areas of strong shock interactions and strong viscous interactions such as interacting shock waves on a leading edge, shock/boundary layer interactions, corner flows, etc., is required. In general, ground based test facilities can provide only limited data for the expected flight conditions, at a considerable cost, and thus designers must depend heavily on analytically predicted aerothermal loads. The analytical methodologies for these predictions must be accurate (to within some predetermined measure), robust, computationally economical, and geometry independent.

For a designer, the prediction of aerothermal loads of hypersonic structures is one of the most challenging problems in computational fluid dynamics. It requires flow analysis and shock prediction at very high Mach numbers, as well as a realistic calculation of aerothermal loads. These tasks required, in general, advanced computational strategies, extremely accurate discretization techniques, and powerful postprocessing capabilities.

During the first two phases of this project we have developed, at the Computational Mechanics Company, an implicit/explicit, anisotropic *h-p* finite element methodology for the analysis of high speed flows and the prediction of aerothermal loads. The basic idea of the *h-p* version of the finite element method is to combine local mesh refinement (an *h*-method) with anisotropic polynomial enrichment (a *p*-method) in order to achieve convergence rates not attainable with fixed mesh methods or with any of the above methods applied separately. In practical terms, the *h-p* method means maximum numerical accuracy at a minimal computational cost.

The remainder of this report summarizes the results of the first and second phases of a four year research and development effort oriented toward the resolution of several theoretical and computational issues related to the above problems. In Section 2 the basic formulation of the Navier-Stokes equations which governs the compressible viscous flow is presented. This is followed by a detailed discussion of a general family of implicit Taylor Galerkin algorithms for solving the compressible flow equations, and a review of the numerical boundary conditions associated with the algorithms. The final part of this section provides a brief discussion about artificial dissipation mechanisms which may be more appropriate for highly distorted elements or elements with high aspect ratios. The Taylor-Galerkin formulation

and the general boundary condition treatment presented in Section 2 is excerpted from our published work on these results (e.g. [8, 10, 31]). The next section, Section 3, provides a detailed review of the *h-p* finite element methodology. Included in this review is a discussion of the data structure, hierarchical shape functions, constrained approximation, and assumptions and restrictions which have been incorporated into the *h-p* formulation. The work reported in Sections 3.3–3.11 is based on extensions of the general *h-p* adaptivity methods of Oden, Demkowicz, and Rachowicz, originally published in 1989 (see [9, 25]) and expanded and generalized to apply to thermomechanical load predictions in this project. Section 4 follows with a discussion of the adaptive strategy which is used in conjunction with the mesh modification algorithms. Here the error estimates used to drive the adaptive package are outlined and the procedure for adapting the computational mesh is presented. Section 5 follows with a detailed discussion of the implicit/explicit methodologies that have been implemented within the context of the two-dimensional code. In particular, in this section we review the basic concepts associated with using implicit/explicit methods and how one would select implicit end explicits zones within the computational domain. This section concludes with a brief overview of other computational procedures also required to efficiently implement an implicit/explicit methodology.

The contents of this report includes a summary of the efforts completed during both the first and second phases of the development effort. During the second phase of the project, our efforts have focused on implicit/explicit methods, directional enrichment techniques and algorithms, and optimal artificial dissipation mechanisms. The details of this effort are provided in sections 2.2, 4.2, 5, and 7. The other basic sections are essentially the same as in previous reports except for various corrections.

The next section, Section 6, presents some non-standard algorithms used in the element calculations and the postprocessing module. This is followed by Section 7, which presents the results of the numerical test cases that have been run over the course of this project. This includes both simple test problems used for verification purposes as well as the benchmark problems supplied by NASA-Langley. The final section provides a brief discussion of some possible directions for future research and development in the area of *h-p* adaptive methods for hypersonic flows.

2 Formulation of the Governing Equations and Numerical Algorithms

This section provides a summary of the basic field equations and numerical algorithms that have been used to model the complex flow phenomena that is encountered in hypersonic flight conditions. We begin this section with a standard review of the basic field equations of compressible gas dynamics which are expressed in conservation form. These equations are then nondimensionalized in the usual fashion (see reference [1]) using the free stream density, velocity, Mach number, Reynolds number, etc. This procedure results in a system of evolution equations, for a set of nondimensionalized conservation variables, which assume essentially the same form as the original governing equations but with new material constants. The next two sections present the details of two algorithms for integrating the Navier-Stokes equations forward in time. The first algorithm is a one–step implicit/explicit method which is based on the general family of Taylor-Galerkin method with several parameters controlling the actual implicitness of the scheme. The second method is a two–step method which separates the Euler fluxes from the viscous contributions, resulting in an inviscid convection step and a viscous diffusion step. The next subsection summarizes the various classes of boundary conditions that arise from these two approaches and the final section discusses artificial dissipation mechanisms appropriate for *h-p* adaptive methods. (As a matter of notational simplicity all of the formulations and algorithms presented in this section are limited to the two–dimensional case.)

2.1 Notation and Formulation of the Equations

The compressible viscous flow of a calorically perfect gas is governed by the Navier-Stokes equations which may be conveniently written as

$$\boldsymbol{U}_{,t} + \boldsymbol{F}_i^C(\boldsymbol{U})_{,i} = \boldsymbol{F}_i^V(\boldsymbol{U})_{,i} \tag{2.1}$$

where $\boldsymbol{U}$ is the vector of conservation variables defined by

$$\boldsymbol{U} = \{\rho, m_1, m_2, e\}^T \tag{2.2}$$

and $\boldsymbol{F}_i^C(\boldsymbol{U})$ and $\boldsymbol{F}_i^V(\boldsymbol{U})$ are the inviscid and viscous fluxes, respectively. The indices i in (2.1) refer to the axis of a Cartesian coordinate system, a comma denotes partial differentiation, and the summation convention is applied. The Eulerian fluxes $\boldsymbol{F}_i^C(\boldsymbol{U})$ are defined as

$$
\begin{aligned}
\boldsymbol{F}_1^C(\boldsymbol{U}) &= \left(m_1, \frac{m_1^2}{\rho} + p, \frac{m_1 m_2}{\rho}, (e+p)\frac{m_1}{\rho}\right)^T \\
\boldsymbol{F}_2^C(\boldsymbol{U}) &= \left(m_2, \frac{m_1 m_2}{\rho}, \frac{m_2^2}{\rho} + p, (e+p)\frac{m_2}{\rho}\right)^T
\end{aligned}
\tag{2.3}
$$

Here ρ is the mass density, $m_i = \rho u_i$ are the momentum components (with u_i the velocity vector components), e is the total energy per unit volume, and p is the thermodynamic pressure. Under the assumption that the fluid behaves as a perfect gas, the constitutive equation relating the pressure to the internal energy is given by

$$
p = (\gamma - 1)\iota \tag{2.4}
$$

where ι is the internal energy per unit volume

$$
\iota = e - \frac{1}{2}\rho(u_1^2 + u_2^2) \tag{2.5}
$$

and γ is the ratio of specific heats

$$
\gamma = \frac{c_p}{c_v} \tag{2.6}
$$

In this expression C_p is the specific heat at constant pressure and C_v is the specific heat at constant volume.

The viscous fluxes $\boldsymbol{F}_i^V(\boldsymbol{U})$ are defined by

$$
\begin{aligned}
F_1^V(\boldsymbol{U}) &= (0, \tau_{11}, \tau_{12}, \tau_{1j}u_j + q_1) \\
F_2^V(\boldsymbol{U}) &= (0, \tau_{21}, \tau_{22}, \tau_{2j}u_j + q_2)
\end{aligned}
\tag{2.7}
$$

with the viscous stresses τ_{ij} related to the velocity gradient through the usual constitutive relation for a Newtonian fluid

$$
\tau_{ij} = \mu(u_{i,j} + u_{j,i}) + \lambda u_{k,k}\delta_{ij} \tag{2.8}
$$

and the heat flux is defined by Fourier's law

$$
q_i = \kappa T_{,i} \tag{2.9}
$$

Here μ and λ are the dynamic viscosity and second viscosity coefficient, respectively, δ_{ij} is the Kronecker delta, T is the temperature, and κ is the heat conduction coefficient.

In addition to the quantities defined above, we also introduce the Reynolds number Re_L and the Prandtl number Pr

$$Re_L = \frac{\rho V L}{\mu} \tag{2.10}$$

$$Pr = \frac{c_p \mu}{\kappa} \tag{2.11}$$

where L is the referential or characteristic length. These quantities will be used in the next section where a nondimensionalized form of the governing equations is discussed.

As a final note, we will assume throughout this report that the Stokes relation is in effect

$$3\lambda + 2\mu = 0 \tag{2.12}$$

and Southerland's law relating the temperature to the dynamic viscosity holds

$$\mu = C_1 \frac{T^{3/2}}{T + C_2} \tag{2.13}$$

where C_1 and C_2 are constants for a given gas.

2.1.1 Nondimensional Form of the Navier-Stokes Equations

Following the procedure outlined by Anderson [1] we introduce the following scaling parameters

- ρ_∞ - free stream density
- V_∞ - free stream velocity
- T_∞ - free stream temperature
- μ_∞ - free stream dynamic viscosity
- L - referential or characteristic length

Using these scaling parameters, a number of nondimensional quantities may be defined

$$
\begin{aligned}
x_i^* &= \frac{x_i}{L} \\
t^* &= \frac{V_\infty t}{L} \\
u_i^* &= \frac{u_L}{V_\infty} \\
\mu^* &= \frac{\mu}{\mu_\infty} \\
\rho^* &= \frac{\rho}{\rho_\infty} \\
p^* &= \frac{p}{\rho_\infty V_\infty^2} \\
T^* &= \frac{T}{T_\infty} \\
\iota^* &= \frac{\iota}{\rho_\infty V_\infty^2}
\end{aligned}
\tag{2.14}
$$

where the nondimensional variables are denoted with an asterisk, t^* is the nondimensionalized time, and x_i^* are the nondimensionalized coordinates. Applying this nondimensionalization to the compressible Navier-Stokes equations, one obtains a set of equations which assume essentially the same form as the original formulas provided that new material constants are defined as follows:

$$
\begin{aligned}
\tilde{\mu} &= \frac{\mu^*}{Re_L}\,;\, \tilde{\lambda} = \frac{\lambda^*}{Re_L} \\
\tilde{c}_v &= [\gamma(\gamma-1)M_\infty^2]^{-1}
\end{aligned}
\tag{2.15}
$$

In equation (2.15), M_∞ is the free stream Mach number which is given by the relation

$$
M_\infty = \frac{V_\infty}{\sqrt{\gamma(\gamma-1)c_v T_\infty}} \tag{2.16}
$$

Throughout this report, asterisks and tildes are omitted and it is understood that the nondimensionalized forms of the governing equations are in use.

This section is concluded with a list of some nondimensional quantities of interest:

- pressure coefficient

$$C_p = \frac{p - p_\infty}{0.5\rho_\infty V_\infty^2} = 2(p^* - \frac{1}{\gamma M_\infty^2}) \tag{2.17}$$

- skin friction coefficient

$$C_f = \frac{\tau_s}{0.5\rho_\infty V_\infty^2} = 2\tau_s^* \tag{2.18}$$

where the nondimensional shear stress τ_s^* is calculated using the nondimensional form of (2.8) (with $\tilde{\mu}$ and $\tilde{\lambda}$ replacing μ and λ, respectively)

- heat flux coefficient

$$C_h = \frac{q}{0.5\rho_\infty V_\infty^3} = 2q^* \tag{2.19}$$

where

$$q = q_i n_i \text{ and } q^* = q_i^* n_i \tag{2.20}$$

and where $\boldsymbol{n} = (n_i)$ denotes a unit normal vector. The nondimensional heat flux vector q_i^* is evaluated using formula (2.9), the nondimensional temperature T^* and the coefficient of conductivity

$$\tilde{\kappa} = \frac{\gamma \tilde{c}_v \tilde{\mu}}{Pr} \tag{2.21}$$

2.2 Taylor-Galerkin Algorithms

This section presents two Taylor-Galerkin algorithms for integrating the Navier-Stokes equations forward in time. The first algorithm is a implicit/explicit one–step approach where all of the components of the solution are handled simultaneously. The second approach is a rather unique two–step approach whereby the Navier-Stokes equations have been split into a convective part and a diffusive part and are subsequently combined to advance the solution in time.

2.2.1 One–Step Taylor-Galerkin Algorithm

A general family of implicit Taylor-Galerkin methods is summarized in this section. These methods can be made explicit or implicit by the appropriate selection of implicit parameters (see below). The details concerning the theoretical formulation can be found in [27] and references therein.

Given a domain $\Omega \subset \mathbb{R}^N$ the compressible Navier-Stokes equations are characterized by a system of conservation laws of the form

$$\boldsymbol{U}_{,t} + \boldsymbol{F}^C_{i,i} = \boldsymbol{F}^V_{i,i} \qquad \boldsymbol{x} \in \Omega,\ t > 0 \tag{2.22}$$

accompanied by an initial condition

$$\boldsymbol{U}(\boldsymbol{x}, 0) = \boldsymbol{U}_0(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega \tag{2.23}$$

and appropriate boundary conditions.

Second-Order Taylor Expansion in Time

As a starting point, let us assume that the solution $\boldsymbol{U}^n$ at time step t^n is known and the solution $\boldsymbol{U}^{n+1}$ at time t^{n+1} is to be calculated. Formally, the values of the solution at times t^n and time t^{n+1} can be expressed by the second–order Taylor series expansion about an arbitrary time $t^{n+\alpha}$ where α is an implicitness parameter with values between zero and one:

$$\begin{aligned} \boldsymbol{U}^{n+1} &= \boldsymbol{U}^{n+\alpha} + (1-\alpha)\Delta t \boldsymbol{U}_t^{n+\alpha} + (1-\alpha)^2 \frac{\Delta t^2}{2} \boldsymbol{U}_{tt}^{n+\alpha} + O(\Delta t^3) \\ \boldsymbol{U}^{n} &= \boldsymbol{U}^{n+\alpha} - \alpha \Delta t \boldsymbol{U}_t^{n+\alpha} + \alpha^2 \frac{\Delta t^2}{2} \boldsymbol{U}_{tt}^{n+\alpha} + O(\Delta t^3) \end{aligned} \tag{2.24}$$

By subtracting these two formulas a formula is obtained for an increment of the solution between steps n and $n+1$:

$$\Delta \boldsymbol{U} = \boldsymbol{U}^{n+1} - \boldsymbol{U}^n = \Delta t \boldsymbol{U}_t^{n+\alpha} + (1-2\alpha)\frac{\Delta t^2}{2} \boldsymbol{U}_{tt}^{n+\alpha} + O(\Delta t^3) \tag{2.25}$$

Observing that:

$$\boldsymbol{U}_{tt}^{n+\alpha} = \boldsymbol{U}_{tt}^{n+\beta} + O((\alpha - \beta)\Delta t) \tag{2.26}$$

a second implicitness parameter can be introduced into equation (2.25) while still preserving the second–order accuracy,

$$\Delta \boldsymbol{U} = \Delta t \boldsymbol{U}_t^{n+\alpha} + (1-2\alpha)\frac{\Delta t^2}{2} \boldsymbol{U}_{tt}^{n+\beta} + O(\Delta t^3) \tag{2.27}$$

The next step is to express the quantities evaluated at times $t^{n+\alpha}$ and $t^{n+\beta}$ by quantities evaluated at the basic steps t^n and t^{n+1},

$$\boldsymbol{U}_t^{n+\alpha} = \boldsymbol{U}_t^n + \alpha \Delta \boldsymbol{U}_t + O(\Delta t^2) \tag{2.28}$$

$$\boldsymbol{U}_{tt}^{n+\beta} = \boldsymbol{U}_{tt}^n + \beta \Delta \boldsymbol{U}_{tt} + O(\Delta t^2) \tag{2.29}$$

Substituting these formulas into equation (2.27) yields a two-parameter expansion:

$$\Delta \boldsymbol{U} = \Delta t \left(\boldsymbol{U}_t^n + \alpha \Delta \boldsymbol{U}_t\right) + (1 - 2\alpha)\frac{\Delta t^2}{2}\left(\boldsymbol{U}_{tt}^n + \beta \Delta \boldsymbol{U}_{tt}\right) + O(\Delta t^3) \tag{2.30}$$

Now, following the procedure outlined by Lax and Wendroff [21], the governing equations can be used to replace time derivatives with spatial derivatives. This substitution yields a formula for the first derivatives:

$$\boldsymbol{U}_t = \boldsymbol{F}_{i,i}^V - \boldsymbol{F}_{i,i}^C \tag{2.31}$$

and for the second-order derivatives:

$$\boldsymbol{U}_{tt} = \left[\boldsymbol{R}_{ij}\left(\boldsymbol{F}_{k,k}^V - \boldsymbol{F}_{k,k}^C\right)_{,j} + \boldsymbol{P}_i\left(\boldsymbol{F}_{k,k}^V - \boldsymbol{F}_{k,k}^C\right)\right]_{,i} - \left[\boldsymbol{A}_i\left(\boldsymbol{F}_{k,k}^V - \boldsymbol{F}_{k,k}^C\right)\right]_{,i} \tag{2.32}$$

where $\boldsymbol{A}_i$, $\boldsymbol{P}_i$, and $\boldsymbol{R}_{ij}$ are the Jacobian fluxes

$$\boldsymbol{A}_i = \frac{\partial \boldsymbol{F}_i^C}{\partial \boldsymbol{U}} \qquad \boldsymbol{P}_i = \frac{\partial \boldsymbol{F}_i^V}{\partial \boldsymbol{U}} \qquad \boldsymbol{R}_{ij} = \frac{\partial \boldsymbol{F}_i^V}{\partial \boldsymbol{U}_{,j}} \tag{2.33}$$

The convective terms in equation (2.32) involve spatial derivatives up to fourth order. Limiting this formula to terms with second-order derivatives, which can be effectively handled by C° continuous finite elements, yields the following approximation for the second-order time derivative

$$\boldsymbol{U}_{tt} = \left(\boldsymbol{A}_i \boldsymbol{F}_{k,k}^C\right)_{,i} + O(\mu, k) \tag{2.34}$$

where $O(\mu, k)$ represents a quantity of the order of the viscosity parameters in the Navier-Stokes equations. Substitution of formulas (2.31) and (2.34) into the incremental equations (2.30) gives the implicit formula:

$$\begin{aligned} \Delta \boldsymbol{U} &= \Delta t\left[\left(\boldsymbol{F}_{i,i}^{Vn} - \boldsymbol{F}_{i,i}^{Cn}\right) + \alpha\left(\Delta \boldsymbol{F}_{i,i}^V - \Delta \boldsymbol{F}_{i,i}^C\right)\right] \\ &+ (1 - 2\alpha)\frac{\Delta t^2}{2}\left[\left(\boldsymbol{A}_i \boldsymbol{F}_{k,k}^C\right)_{,i} + \beta \Delta\left(\boldsymbol{A}_i \boldsymbol{F}_{k,k}^C\right)_{,i}\right] + O\left(\Delta t^3\right) + O(\mu, k) O\left(\Delta t^2\right) \end{aligned} \tag{2.35}$$

For the sake of maximum generality, a third implicitness parameter can be introduced by observing that

$$\alpha\Delta \boldsymbol{F}^{V}_{i,i} - \alpha\Delta \boldsymbol{F}^{C}_{i,i} = \gamma\Delta \boldsymbol{F}^{V}_{i,i} - \alpha\Delta \boldsymbol{F}^{C}_{i,i} + (\alpha - \gamma)O(\mu, k)O(\Delta t) \tag{2.36}$$

Substituting this expression into (2.35) yields the three-parameter implicit form for the increments of the conservation vector

$$\begin{aligned} \Delta \boldsymbol{U} &= \Delta t \left[\left(\boldsymbol{F}^{Vn}_{i,i} - \boldsymbol{F}^{Cn}_{i,i} \right) + \gamma\Delta \boldsymbol{F}^{V}_{i,i} - \alpha\Delta \boldsymbol{F}^{C}_{i,i} \right] \\ &+ (1 - 2\alpha)\frac{\Delta t^2}{2} \left[\left(\boldsymbol{A}_i \boldsymbol{F}^{C}_{k,k} + \beta\Delta \left(\boldsymbol{A}_i \boldsymbol{F}^{C}_{k,k} \right)_{,i} \right) \right] \end{aligned} \tag{2.37}$$

Equation (2.37) represents a nonlinear formula for increments of the solution $\boldsymbol{U}$ at a given time step. This formula is nonlinear due to nonlinear dependence of the fluxes and Jacobians on the solution $\boldsymbol{U}$. This equation can be linearized, while still preserving second-order accuracy, with the resulting incremental formula:

$$\begin{aligned} \Delta \boldsymbol{U} &+ \alpha\Delta t \left(\boldsymbol{A}^{n}_{i} \Delta \boldsymbol{U} \right)_{,i} - \gamma\Delta t \left[\left(\boldsymbol{R}^{n}_{ij} \Delta \boldsymbol{U}_{,j} \right)_{,i} + \left(\boldsymbol{P}^{n}_{i} \Delta \boldsymbol{U} \right)_{,i} \right] \\ &- (1 - 2\alpha)\beta \frac{\Delta t^2}{2} \left(\boldsymbol{A}^{n}_{i} \boldsymbol{A}^{n}_{j} \Delta \boldsymbol{U}_{,j} \right)_{,i} \\ &= \Delta t \left(\boldsymbol{F}^{Vn}_{i,i} - \boldsymbol{F}^{Cn}_{i,i} \right) + (1 - 2\alpha) \frac{\Delta t^2}{2} \left(\boldsymbol{A}^{n}_{i} \boldsymbol{F}^{Cn}_{k,k} \right)_{,i} \end{aligned} \tag{2.38}$$

The particular form of this evolution equation used in the finite element approximation corresponds to setting $\alpha = 0$ in (2.38). For this special case one obtains

$$\begin{aligned} \Delta \boldsymbol{U} &- \gamma\Delta t \left[\left(\boldsymbol{R}^{n}_{ij} \Delta \boldsymbol{U}_{,j} \right)_{,i} + \left(\boldsymbol{P}^{n}_{i} \Delta \boldsymbol{U} \right)_{,i} \right] \\ &- \beta \frac{\Delta t^2}{2} \left(\boldsymbol{A}^{n}_{i} \boldsymbol{A}^{n}_{j} \Delta \boldsymbol{U}_{,j} \right)_{,i} \\ &= \Delta t \left(\boldsymbol{F}^{Vn}_{i,i} - \boldsymbol{F}^{Cn}_{i,i} \right) + \frac{\Delta t^2}{2} \left(\boldsymbol{A}^{n}_{i} \boldsymbol{F}^{Cn}_{k,k} \right)_{,i} \end{aligned} \tag{2.39}$$

Weak Formulation

In order to obtain a weak variational formulation of the incremental equation (2.39), we introduce the space of test functions

$$W = \left\{ \boldsymbol{V} = (V_1, V_2 \ldots V_M) \;\; s.t. \; V_i \in H^1(\Omega) \text{ and } V_i = 0 \text{ on } \Gamma_D \right\} \tag{2.40}$$

where M is the number of conservation variables, $H^1(\Omega)$ is the usual Sobolev space of functions with derivatives in $L^2(\Omega)$, and Γ_D is the boundary with specified Dirichlet boundary conditions. After multiplication of the incremental equation (2.39) by an arbitrary test function $\boldsymbol{V}(\boldsymbol{x}) \in W$, integrating over the domain Ω and application of the divergence theorem, the following weak formulation of the problem is obtained:

Find $\boldsymbol{\Delta U} \in W \; s.t. \; \forall \, \boldsymbol{V} \in W$:

$$\begin{aligned}
&\int_\Omega \Big(\boldsymbol{\Delta U} \cdot \boldsymbol{V} + \gamma \Delta t \boldsymbol{R}^n_{ij} \boldsymbol{\Delta U}_{,j} \cdot \boldsymbol{V}_{,i} \\
&+ \gamma \Delta t \boldsymbol{P}^n_i \boldsymbol{\Delta U} \cdot \boldsymbol{V}_{,i} + \left(\beta \frac{\Delta t^2}{2} \boldsymbol{A}^n_i \boldsymbol{A}^n_j \boldsymbol{\Delta U}_{,j} \cdot \boldsymbol{V}_{,i} \right) d\Omega \\
&- \int_{\partial\Omega} \Big(\gamma \Delta t n_i \boldsymbol{R}^n_{ij} \boldsymbol{\Delta U}_{,j} \cdot \boldsymbol{V} \\
&+ \gamma \Delta t n_i \boldsymbol{P}^n_i \boldsymbol{\Delta U} \cdot \boldsymbol{V} + \beta \frac{\Delta t^2}{2} n_i \boldsymbol{A}^n_i \boldsymbol{A}^n_j \boldsymbol{\Delta U}_{,j} \cdot \boldsymbol{V} \Big) dS \\
&= \int_\Omega \left(\Delta t \left(\boldsymbol{F}^{Cn}_i - \boldsymbol{F}^{Vn}_i \right) \cdot \boldsymbol{V}_{,i} - \frac{\Delta t^2}{2} \boldsymbol{A}^n_i \boldsymbol{A}^n_j \boldsymbol{U}_{,j} \cdot \boldsymbol{V}_{,i} \right) d\Omega \\
&+ \int_{\partial\Omega} \left(-\Delta t n_i \left(\boldsymbol{F}^{Cn}_i - \boldsymbol{F}^{Vn}_i \right) \cdot \boldsymbol{V} + \frac{\Delta t^2}{2} n_i \boldsymbol{A}^n_i \boldsymbol{F}^{Cn}_{j,j} \cdot \boldsymbol{V} \right) dS
\end{aligned} \tag{2.41}$$

It can be shown by the selection of appropriate test functions that the solution of this problem is, in the sense of distributions, the solution of the boundary–value problem, together with appropriate boundary conditions. Additional details concerning implicit Taylor-Galerkin methods may be found in references [11,15,22,31].

2.2.2 The Two-Step Algorithm

A second algorithm investigated during the course of this project for solving Navier-Stokes equations is based on a two–step approach [8,10,28]. The method consists of advancing the solution in time by performing interchangably two steps associated with the convection operator $\boldsymbol{E}$ and the diffusion operator $\boldsymbol{H}$ corresponding to inviscid and viscous terms in equation (2.1):

$$\boldsymbol{U}^{n+1} = \boldsymbol{G}(t)\boldsymbol{U}^n \tag{2.42}$$

where $\boldsymbol{G}(t) = \boldsymbol{H}(t)\boldsymbol{E}(t)$. The convection operator $\boldsymbol{E}$(t) is defined by:

$$(\boldsymbol{E}(t)\boldsymbol{U}_0)(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{U}(\boldsymbol{x},t) \tag{2.43}$$

where $\boldsymbol{U}(\boldsymbol{x},t)$ is a solution to Euler equation:

$$\begin{cases} \boldsymbol{U}_{,t} + \boldsymbol{F}_i^C(\boldsymbol{U})_{,i} = 0 \\ \\ \boldsymbol{U}(\boldsymbol{x},0) = \boldsymbol{U}_0(\boldsymbol{x}). \end{cases} \tag{2.44}$$

The diffusion operator $\boldsymbol{H}(t)$ is defined by:

$$(\boldsymbol{H}(t)\boldsymbol{U}_0)(\boldsymbol{x}) \stackrel{\text{def}}{=} \boldsymbol{U}(\boldsymbol{x},t) \tag{2.45}$$

where $\boldsymbol{U}(\boldsymbol{x},t)$ is a solution to:

$$\begin{cases} \boldsymbol{U}_{,t} = F_i^V(\boldsymbol{U})_{,i} \\ \\ \boldsymbol{U}(\boldsymbol{x},0) = \boldsymbol{U}_0(\boldsymbol{x}). \end{cases} \tag{2.46}$$

Problems (2.44) and (2.46) must be augmented by appropriate boundary conditions, a detailed discussion of which is given in the following section. It should be mentioned that a different composition of operators $\boldsymbol{H}(t)$ and $\boldsymbol{E}(t)$ gives a three-step procedure of the form

$$\boldsymbol{G}(t) = \boldsymbol{H}(\frac{t}{2})\boldsymbol{E}(t)\boldsymbol{H}(\frac{t}{2}) \tag{2.47}$$

which is second order accurate while our two-step procedure is only first order accurate in time. This however is not of our concern since we are interested only in steady-state solutions.

In the numerical implementations, exact operators $\boldsymbol{E}(t)$ and $\boldsymbol{H}(t)$ are replaced by their discrete approximations. The motivation of applying this two-step approach to solving Navier-Stokes equation was that different solvers or even different spatial approximations could be used for the Euler and viscous steps. We may take advantage of this flexibility especially in modeling boundary phenomena: solving, for instance, the convection step with a specialized and very efficient Euler solver and linear approximation, while using very accurate higher order approximation in boundary layer zones in the viscous step.

The Euler (Convection) Step

In this work, we approximate the Euler step (2.44) by a second order Taylor-Galerkin scheme while the viscous step (2.46) is approximated by a first order finite difference approximation.

This reduces the problem to solving at each time step elliptic-like boundary value problems for which we employ our *h-p* finite element method for the spatial approximation.

A Taylor-Galerkin scheme for the Euler equations is obtained by simply setting $\boldsymbol{F}_i^V$ and the various derivatives of $\boldsymbol{F}_i^V$ to zero in equations (2.39) and (2.41). For completeness, this procedure is summarized below.

Given a domain $\Omega \subset \mathbb{R}^N$, the compressible Euler equations are characterized by a system of conservation laws of the form

$$\boldsymbol{U}_{,t} + \boldsymbol{F}_i^C(\boldsymbol{U})_{,i} = \boldsymbol{0} \quad \boldsymbol{x} \in \Omega, \mathrm{t} > 0 \tag{2.48}$$

accompanied by an initial condition

$$\boldsymbol{U}(\boldsymbol{x}, 0) = \boldsymbol{U}_0(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega \tag{2.49}$$

and by appropriate boundary conditions.

Now taking equation (2.39) and setting $\boldsymbol{R}_{ij}$, $\boldsymbol{P}_i$ and $\boldsymbol{F}_i^V$ to zero one obtains an evolution equation for the conservation variables

$$\begin{aligned} \Delta \boldsymbol{U} \;-\; & \beta \frac{\Delta t^2}{2} \left(\boldsymbol{A}_i^n \boldsymbol{A}_j^n \Delta \boldsymbol{U}_{,j} \right)_{,i} = \\ -\; & \Delta t \boldsymbol{F}_{i,i}^{Cn} + \frac{\Delta t^2}{2} \left(\boldsymbol{A}_i^n \boldsymbol{F}_{k,k}^{Cn} \right)_{,i} \end{aligned} \tag{2.50}$$

The corresponding weak formulation follows from (2.41) as

Find $\Delta \boldsymbol{U} \in W$ *s.t.* $\forall\, \boldsymbol{V} \in W$

$$\begin{aligned} & \int_\Omega \left(\Delta \boldsymbol{U} \cdot \boldsymbol{V} + \beta \frac{\Delta t^2}{2} \boldsymbol{A}_i^n \boldsymbol{A}_j^n \Delta \boldsymbol{U}_{,j} \cdot \boldsymbol{V}_{,i} \right) d\Omega \\ & - \int_{\partial\Omega} \left(\beta \frac{\Delta t^2}{2} n_i \boldsymbol{A}_i^n \boldsymbol{A}_j^n \Delta \boldsymbol{U}_{,j} \cdot \boldsymbol{V} \right) dS \\ & = \int_\Omega \left(\Delta t \boldsymbol{F}_i^{Cn} \cdot \boldsymbol{V}_{,i} - \frac{\Delta t^2}{2} \boldsymbol{A}_i^n \boldsymbol{A}_j^n \boldsymbol{U}_{,j} \cdot \boldsymbol{V}_{,i} \right) d\Omega \\ & + \int_{\partial\Omega} \left(-\Delta t n_i \boldsymbol{F}_i^{Cn} \cdot \boldsymbol{V} + \frac{\Delta t^2}{2} n_i \boldsymbol{A}_i^n \boldsymbol{A}_j^n \boldsymbol{U}_{,j} \boldsymbol{V} \right) dS \end{aligned} \tag{2.51}$$

The Taylor-Galerkin method for solving the Euler step is unconditionally stable for $\beta \geq 0.5$ *independently of the approximation in the space variables.* The second-order terms present on the left-hand side modify the L^2-projection and contribute to the stabilization of the method. Additional details are given in [8].

The Viscous (Diffusion) Step

The second step in the two–step method is the viscous or diffusion step defined by equation (2.46). In this step, the density remains unchanged and evolution equations for the momentum and energy may be fully decoupled provided that the boundary condition for the momentum equations can be formulated so that they do not contain energy terms. Under this assumption, one arrives at a system of two equations to be solved simultaneously for the momentum components

$$\frac{\partial m_i}{\partial t} = \tau_{ij,j} \tag{2.52}$$

and a single scalar equation to solve for the total energy

$$\frac{\partial e}{\partial t} = (\tau_{ij} u_j)_{,i} + q_{i,i} \tag{2.53}$$

As a starting point in the solution of the momentum and energy equations, we introduce a Taylor series expansion of the conservation vector $\boldsymbol{U}$ about an arbitrary time $t + \beta\Delta t$

$$\boldsymbol{U}(t + \Delta t) - \beta\Delta t \boldsymbol{U}_{,t}(t + \Delta t) = \boldsymbol{U}(t) + (1 - \beta)\Delta t \boldsymbol{U}_{,t}(t) + O(\Delta t^2) \tag{2.54}$$

where β is a constant between zero and one. Again using the Lax-Wendroff procedure for replacing the time derivatives by spatial derivatives one obtains

$$m_j^{n+1} - \beta\Delta t \tau_{ij,i}^{n+1} = m_j^n + (1 - \beta)\Delta t \tau_{ij,i}^n \tag{2.55}$$

and

$$\begin{aligned} e^{n+1} - \beta\Delta t \sum_{i=1}^{2} \left(\tau_{ij}^{n+1} u_j^{n+1} + \kappa T_{,i}^{n+1} \right)_{,i} \\ = e^n + (1 - \beta)\Delta t \sum_{i=1}^{2} \left(\tau_{ij}^n u_j^n + \kappa T_{,i}^n \right)_{,i} \end{aligned} \tag{2.56}$$

The procedure for solving the system of equations then becomes: solve the inviscid step using the implicit Taylor-Galerkin method outlined in (2.51), next solve the momentum step defined by equation (2.55), and finally solve the energy equation for the temperature. Combining these results as indicated by equation (2.42) advances the solution of the Navier-Stokes equation found in time by Δt.

The variational formulation for the momentum and energy equations are obtained using exactly the same procedure that was used for the convective or inviscid step. Performing these steps one arrives at the following variational formulations:

Find m_j^{n+1} such that

$$\begin{aligned}
&\int_\Omega m_i^{n+1} V_i d\Omega + \beta\Delta t \int_\Omega \tau_{ij}^{n+1} V_{i,j} d\Omega \\
&- \beta\Delta t \int_{\partial\Omega} \tau_{ij}^{n+1} n_j V_i dS = \int_\Omega m_i^n V_i d\Omega \\
&- (1-\beta)\Delta t \int_\Omega \tau_{ij}^n V_{i,j} d\Omega + (1-\beta)\Delta t \int_{\partial\Omega} \tau_{ij}^n n_j V_i
\end{aligned} \tag{2.57}$$

for every V_i

and

Find e^{n+1} such that

$$\begin{aligned}
&\int_\Omega e^{n+1} V d\Omega + \beta\Delta t \int_\Omega \kappa T_{,i}^{n+1} V_{,i} d\Omega \\
&- \beta\Delta t \int_{\partial\Omega} \kappa T_{,i}^{n+1} n_i V ds = \\
&\int_\Omega e^n V d\Omega - (1-\beta)\Delta t \int_\Omega \kappa T_{,i}^n V_{,i} d\Omega \\
&+ (1-\beta)\Delta t \int_\Omega \kappa T_{,i}^n n_i V dS \\
&+ \Delta t \int_\Omega (\beta \tau_{ij}^{n+1} u_j^{n+1} + (1-\beta)\tau_{ij}^n u_j^n V_{,i} d\Omega
\end{aligned} \tag{2.58}$$

for every V

Note that in the variational formulation of the energy equation one obtains a volume integral which is a function of the stress and velocity at time $t + \Delta t$. By allowing the viscosity parameters λ and μ to lag a time step, we are able to first solve the momentum equation for the velocity components and then explicitly evaluate this final term.

2.3 Boundary Conditions

This section presents a general overview of COMCO's approach to prescribing boundary conditions for the Navier-Stokes equations. This approach is based upon a linearized stabil-

$$A_0 = \begin{bmatrix} \frac{1}{\rho}[\psi^2+\gamma] & -\psi\frac{u}{\iota} & -\psi\frac{v}{\iota} & \frac{1}{\iota}(\psi-1) \\ & \frac{1}{\iota}\left(1+\rho\frac{u^2}{2}\right) & \rho\frac{uv}{\iota^2} & -\rho\frac{u}{\iota^2} \\ \text{Sym.} & & \frac{1}{\iota}\left(1+\rho\frac{v^2}{\iota}\right) & -\frac{\rho v}{\iota^2} \\ & & & \frac{\rho}{\iota^2} \end{bmatrix}$$

where $\psi = \rho\dfrac{u^2+v^2}{2\iota}$.

Figure 2.1: The symmetrizer.

ity analysis of the Navier-Stokes equations which results in the following entropy stability condition that must be satisfied on the boundary

$$\int_{\partial\Omega}\left(\frac{1}{2}\delta U^T A_0 A_n \delta U - \delta U^T A_0 \delta t_n\right) ds \geq 0 \tag{2.59}$$

In this equation, δU is the variation in the conservation vector, A_n is the normal Jacobian matrix defined by $A_n = A_i n_i$, A_0 is the symmetrizer of the Navier-Stokes equations shown in Fig. 2.1, and δt_n is the variation in the normal viscous flux. Additional information on the symmetrizer and stability conditions (2.61) can be found in references [8,9].

2.3.1 Boundary Condition for the One–Step Algorithm

We begin our discussion of boundary conditions for the implicit one–step Taylor-Galerkin methods by quoting a result of Strikverda [29], which specifies the number of boundary

conditions necessary for well-posedness of the linearized Euler and Navier-Stokes equations. These results are summarized for two-dimensional problems in Table 2.1.

Table 2.1

Type of Boundary	Euler	Navier-Stokes
supersonic inflow	4 ess	4 ess
subsonic inflow	3 ess	3 ess + 1 nat
subsonic outflow	1 ess	1 ess + 2 nat
supersonic outflow	0	0 ess + 3 nat
no-flow	1 ess	1 ess + 2 nat
solid wall		
—isothermal	—	3 ess
—heat flux	—	2 ess + 1 nat

In this table, "ess" denotes the essential boundary condition and "nat" denotes natural boundary conditions. The essential conditions are to be imposed on the characteristic variables rather than the conservation variables. It is important to note that the numbers presented in the table are true for problems that are not regularized. If artificial diffusion is built into the algorithm or added explicitly, natural boundary conditions should be imposed on these terms even for Euler problems. Moreover, since artificial diffusion can affect all conservation variables, the number of natural boundary conditions for these terms should actually be one more than for the (nonregularized) Navier-Stokes equations.

Before launching into a full discussion of the various classes of boundary conditions, it is useful to first recast the boundary integrals in terms of the characteristic variables (Riemann invariants). In the variational formulation a typical boundary term is of the form

$$\boldsymbol{A}_i n_i \Delta \boldsymbol{U} \cdot \boldsymbol{V} = \boldsymbol{A}_n \Delta \boldsymbol{U} \cdot \boldsymbol{V} \tag{2.60}$$

where $\boldsymbol{A}_n$ is a nonsymmetric matrix. The matrix $\boldsymbol{A}_n$ can formally be represented with respect to its own eigenbasis as

$$\boldsymbol{A}_n = \sum_{\alpha=1}^{M} \lambda_\alpha (\boldsymbol{c}_\alpha \otimes \boldsymbol{b}_\alpha) \tag{2.61}$$

where $\boldsymbol{b}_\alpha$ and $\boldsymbol{c}_\alpha$ are the left and right eigenvectors, respectively. The eigenvalues λ_α for the two–dimensional case are

$$\begin{aligned} \lambda_1 &= u_n - c \\ \lambda_2 &= u_n \\ \lambda_3 &= u_n \\ \lambda_4 &= u_n + c \end{aligned} \tag{2.62}$$

where u_n is the velocity normal to the boundary and c is the speed of sound. The expressions for eigenvectors $\boldsymbol{b}_\alpha$ and $\boldsymbol{c}_\alpha$ can be found in references [8,12,29]. Note that a positive value of λ_α means that the corresponding characteristic exits the domain across a given boundary, while negative values of λ_α correspond to signals entering the domain. As a general rule, the characteristic variables corresponding to characteristics entering the domain (negative λ_α) need to be specified as the essential boundary conditions, while the characteristic variables exiting the domain are continued across the boundary from the interior.

The characteristic variables ΔU_α are defined as components of the vector $\Delta \boldsymbol{U}$ in the eigenbasis of $\boldsymbol{A}_n$ so that

$$\begin{aligned} \boldsymbol{\Delta U} &= (\boldsymbol{\Delta U} \cdot \boldsymbol{b}_\alpha)\boldsymbol{c}_\alpha = \Delta U_\alpha \boldsymbol{c}_\alpha \\ \boldsymbol{V} &= (\boldsymbol{V} \cdot \boldsymbol{c}_\alpha)\boldsymbol{b}_\alpha = V_\alpha \boldsymbol{b}_\alpha \end{aligned}$$

With these definitions, the boundary formula (2.60) can be presented in terms of characteristic variables as:

$$\boldsymbol{A}_n \boldsymbol{\Delta U} \cdot \boldsymbol{V} = \sum_{\alpha=1}^{M} \lambda_\alpha V_\alpha \Delta U_\alpha \tag{2.63}$$

The above representation is very useful in the formulation of essential boundary conditions.

In the following sections the formulation of the boundary conditions for various boundary types is discussed. Additional details can be found in [12,13,25] and references therein.

Supersonic Inflow

On a supersonic inflow boundary, the values of all the characteristic variables (thus also of all the conservation variables) are specified as the upstream values. Formally, this means that

$$\boldsymbol{U} = \overline{\boldsymbol{U}} \tag{2.64}$$

or, in incremental form,

$$\boldsymbol{\Delta U} = \overline{\boldsymbol{U}} - \boldsymbol{U}^n \text{ on } \partial\Omega_I$$

In practical applications these conditions are enforced by the penalty method, which is obtained by adding to the variational formulation a term

$$\frac{1}{\epsilon}\int_{\partial\Omega_I}\left[\boldsymbol{\Delta U}-\left(\overline{\boldsymbol{U}}-\boldsymbol{U}^n\right)\right]\cdot\boldsymbol{V}ds$$

ϵ being a small parameter.

Supersonic Outflow

On a supersonic outflow boundary, all characteristic variables propagate along characteristics so as to be continued from the interior of the domain and no essential boundary conditions are imposed. The explicit algorithm application of supersonic outflow boundary conditions is achieved simply by calculation of the boundary integrals. In the implicit algorithm, natural boundary conditions are imposed on viscous and second-order terms.

Natural boundary conditions on viscous terms are imposed by observing that the viscous boundary terms on the left-hand side of the variational equation (2.41) can be interpreted as

$$\begin{aligned}\boldsymbol{K}^V_{IJ}\boldsymbol{\Delta U}_J &= \int_{\partial\Omega_o}-\gamma\Delta t\left(\boldsymbol{R}^n_{ij}n_i\Psi_{J,j}\Psi_I+\boldsymbol{P}_in_i\Psi_I\Psi_J\right)\boldsymbol{\Delta U}_Jds\\ &= \int_{\partial\Omega_o}-\gamma\Delta t\boldsymbol{\Delta F}^V_n\Psi_Ids\end{aligned}$$

where the components of $\boldsymbol{\Delta F}^V_n$ are

$$\boldsymbol{\Delta F}^V_n=\{0,\Delta\sigma_{1n},\Delta\sigma_{2n},\Delta q_n\}^T$$

and $\boldsymbol{\Psi}(x)$ are the shape functions. In order to formally impose natural boundary conditions, the above terms are transferred to the right-hand side with prescribed values of $\boldsymbol{\Delta F}^V_n$, so that the new right-hand side is

$$\widetilde{\boldsymbol{R}}_I=\boldsymbol{R}_I+\int_{\partial\Omega_o}\gamma\Delta t\overline{\boldsymbol{\Delta F}}^V_n\Psi_Ids$$

Note that since the mass flux due to viscous terms is identically zero, this procedure actually imposes only three natural boundary conditions (in two dimensions).

The choice of the actual conditions is somewhat arbitrary. Currently two options are implemented, namely,

- zero change of flux at the time step (frozen viscous flux):

$$\overline{\boldsymbol{\Delta F}}^V_n=0$$

- zero total viscous flux, enforced by:

$$\overline{\Delta \boldsymbol{F}}_n^V = 0 - \boldsymbol{F}_n^{Vn}$$

Obviously, on an outflow boundary adjacent to a solid wall the viscous fluxes are not zero and the first procedure is more appropriate.

In order to ensure well-posedness of the problem, proper natural boundary conditions should also be imposed on the second-order terms. Analogously as in the viscous case, second-order terms on the left-hand side of equation (2.41) can be transformed to the form:

$$\boldsymbol{K}_{IJ}^D \Delta \boldsymbol{U}_J = \int_{\partial\Omega_o} -\beta \frac{\Delta t^2}{2} \Psi_I \left(\boldsymbol{A}_n \Delta \boldsymbol{F}_{j,j}\right) ds$$

This term has no simple physical interpretation and therefore the selection of the natural boundary condition to be imposed is somewhat difficult. The procedure adopted by Demkowicz, Oden, and Rachowicz [8] is to decompose the above term into components normal and tangential to the boundary and impose boundary conditions only on the normal term (symmetry boundary conditions). In this work, a slightly different procedure is applied, according to which the whole term is transferred to the right-hand side with certain prescribed values. This corresponds to imposing natural boundary conditions on

$$\boldsymbol{A}_n \Delta \boldsymbol{F}_{j,j} = \boldsymbol{A}_n \boldsymbol{A}_j \Delta \boldsymbol{U}_{,j}$$

The actual boundary condition applied is to set the total value of this term to zero, so that the prescribed value at the time step is

$$\boldsymbol{A}_n \boldsymbol{A}_j \Delta \boldsymbol{U}_{,j} = 0 - \boldsymbol{A}_n \boldsymbol{A}_j \boldsymbol{U}_{,j}^n \tag{2.65}$$

An interpretation of this condition can be obtained by observing that for Euler problems, the enforced condition is $\boldsymbol{A}_n \dot{\boldsymbol{U}} = 0$, or in terms of characteristic variables,

$$\lambda_\alpha \dot{U}_\alpha \boldsymbol{b}_\alpha = 0 \qquad \alpha = 1, \ldots, M$$

Since, on the supersonic outflow, all the eigenvalues satisfy $\lambda_\alpha > 0$, this condition means that the characteristic variables do not change in time as they exit the interior across the boundary.

A somewhat more appealing interpretation can be presented for the simple two-dimensional advection equation

$$\dot{\boldsymbol{U}} = a_i \boldsymbol{U}_{,i} \qquad i = 1, 2$$

for which the characteristics are straight lines defined in space-time by the vector $\boldsymbol{c} = \{a_1, a_2, 1\}$. The natural boundary condition corresponding to (2.65) is:

$$n_i a_i a_j \boldsymbol{U}_{,j} = 0$$

or equivalently:

$$a_n \langle \boldsymbol{DU}, \boldsymbol{a} \rangle = 0$$

where $\langle \boldsymbol{DU}, \boldsymbol{a} \rangle$ denotes the directional derivative of $\boldsymbol{u}$ in the direction of $\boldsymbol{a}$. This means that on the outflow boundary ($a_n > 0$) $\boldsymbol{U}$ must be constant along advection lines.

On a contour plot, this forces contours of $\boldsymbol{U}$ to be parallel to advection lines on the boundary. It can be observed that the condition applied by Demkowicz, Oden, and Rachowicz [8] causes derivatives of $\boldsymbol{U}$ along the normal to the boundary $\boldsymbol{n}$ to be zero, which corresponds to contour lines normal to the boundary.

Subsonic Inflow and Outflow

On a subsonic inflow or outflow boundary, essential boundary conditions are imposed only on the characteristic variables corresponding to negative eigenvalues λ_α. For each of these terms, the corresponding condition is

$$\begin{aligned} \Delta U_\alpha &= \Delta \boldsymbol{U} \cdot \boldsymbol{b}_\alpha \\ &= \left(\bar{\boldsymbol{U}} - \boldsymbol{U}^n \right) \cdot \boldsymbol{b}_\alpha \end{aligned}$$

where the prescribed far-field values of the conservation variables are denoted by $\bar{\boldsymbol{U}}$. The penalty term enforcing essential boundary conditions on the increments of selected characteristic variables ΔU_α is of the form

$$\boldsymbol{K}_{IJ} = \frac{1}{\epsilon} \int_{\partial\Omega} (\boldsymbol{c}_\alpha \otimes \boldsymbol{b}_\alpha) \Psi_I \Psi_J ds$$

$$\boldsymbol{R}_J = \frac{1}{\epsilon} \int_{\partial\Omega} \left[(\bar{\boldsymbol{U}} - \boldsymbol{U}^n) \cdot \boldsymbol{b}_\alpha \right] \boldsymbol{c}_\alpha \Psi_I ds$$

Nodewise application of these conditions is obtained by replacing the shape functions with Dirac delta functions associated with boundary nodes.

For the characteristic variables with nonnegative eigenvalues, continuation from interior conditions are employed. These conditions for selected characteristic variables involve rather complicated formulas. Thus, for practical applications it is better to observe that, since the penalty procedure actually overrides any other conditions, the continuation condition can first be applied to the whole vector of conservation variables (by the supersonic outflow procedure), and then the above penalty method can be applied to selectively enforce essential boundary conditions.

In practical implementations of subsonic outflow boundary conditions, some authors choose to prescribe a value of pressure $\bar{p}$ rather than the value of the characteristic vari-

able. The incremental form of the *pressure boundary condition* is

$$\Delta p = \bar{p} - p^n$$

and can be enforced by a penalty method. The corresponding penalty term in the variational formulation is

$$\frac{1}{\varepsilon}\left[\Delta p - (\bar{p} - p^n)\right] \cdot \delta(\Delta p) \tag{2.66}$$

The pressure increment Δp can be expressed in terms of conservation variables as

$$\Delta p = \frac{\partial p}{\partial \boldsymbol{U}} \boldsymbol{\Delta U} = \boldsymbol{d} \cdot \boldsymbol{\Delta U}$$

where $\boldsymbol{d} = (\gamma - 1)\left\{\frac{e_k}{\rho},\ u_1, u_2, 1\right\}^T$ (in the two-dimensional case). Therefore, the penalty term becomes

$$\frac{1}{\varepsilon}\left[\boldsymbol{d} \cdot \boldsymbol{\Delta U} - (\bar{p} - p^n)\right](\boldsymbol{d} \cdot \boldsymbol{V}) \tag{2.67}$$

where $\boldsymbol{V}$ is a test function. The corresponding stiffness matrix and right-hand side are then

$$\boldsymbol{K}_{IJ} = \frac{1}{\varepsilon}\int_{\partial\Omega} \boldsymbol{d} \otimes \boldsymbol{d}\, \Psi_J \Psi_J ds$$

$$\boldsymbol{R}_I = \frac{1}{\varepsilon}\int_{\partial\Omega} \boldsymbol{d}\,(\bar{p} - p^n)\, \Psi_I ds$$

Solid Wall

There are basically two types of solid wall boundaries:

- adiabatic walls with prescribed zero heat flux (M-th component of the viscous flux vector):

$$q_n = F^V_{n(M)} = 0$$

- isothermal walls with specified temperature:

$$T = \overline{T}$$

In addition to the above conditions, zero velocity (zero momentum) conditions are also specified on the solid wall. These conditions are easily enforced by the selective application of a penalty method, similar to the supersonic inflow procedure. The incremental form of the adiabatic condition of zero heat flux is

$$\boldsymbol{\Delta F}^V_{n(M)} = -\boldsymbol{F}^V_{n(M)} \tag{2.68}$$

This natural boundary condition is applied by formally transferring the viscous terms corresponding to the energy equation from the left–hand side of the variational equation (2.41) to the right–hand side and setting the increment of the heat flux according to (2.68). It is of interest to note that since the viscous terms do not directly affect mass fluxes and the momentum equations are overridden by the penalty method, in practice all viscous contributions can be skipped on the left–hand side.

On the isothermal wall the additional boundary condition is a prescribed temperature $\overline{T}$ or, equivalently, a prescribed specific energy $\bar{e}$. Since the kinetic energy is zero on the wall, the above condition can be expressed in terms of conservation variables as:

$$\frac{e}{\rho} = \bar{e}$$

In the incremental form this condition becomes

$$\frac{1}{\rho}\Delta e - \frac{e}{\rho^2}\Delta\rho = (\bar{e} - e^n) \tag{2.69}$$

This condition is imposed via the penalty method. It should be noted that there are available a variety of possible forms of the penalty terms, depending on the form of the test term applied to condition (2.69). One possibility, which appears to be the most natural and yields a symmetric contribution to the stiffness matrix, is obtained by testing equation (2.69) with its own variation:

$$\frac{1}{\epsilon}\left[\left(\frac{1}{\rho}\Delta e - \frac{e}{\rho^2}\Delta\rho\right) - (\bar{e} - e^n)\right] \cdot \left(\frac{1}{\rho}V_{(M)} - \frac{e}{\rho^2}V_{(1)}\right)$$

where $V_{(i)}$ denotes a selected component of a test vector: $V_{(1)}$ for density and $V_{(M)}$ for energy.

This approach leads to two penalty conditions affecting both the continuity and energy equations. Therefore it is not in agreement with the physical situation because, while the solid wall can supply heat to maintain a prescribed termperature, it cannot supply mass for this purpose. For this reason, another form of the penalty term should be used, which enforces a prescribed temperature by altering the energy equation only:

$$\frac{1}{\epsilon}\left[\left(\frac{1}{\rho}\Delta e - \frac{e}{\rho^2}\Delta\rho\right) - (\bar{e} - e^n)\right] \cdot V_{(M)}$$

The corresponding terms in the stiffness matrix and the right–hand side are

$$\begin{aligned} \boldsymbol{K}_{IJ} &= \int_{\partial\Omega_W} \boldsymbol{k}\Psi_I\Psi_J ds \\ \boldsymbol{R}_I &= \int_{\partial\Omega_W} \boldsymbol{r}\Psi_I ds \end{aligned} \tag{2.70}$$

where the kernel matrices $\boldsymbol{k}$ and $\boldsymbol{r}$ are defined (in two dimensions) as:

$$\boldsymbol{k} = \frac{1}{\epsilon}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{e}{\rho} & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{r} = \frac{1}{\epsilon}\begin{bmatrix} 0 \\ 0 \\ 0 \\ \rho(\bar{e} - e^n) \end{bmatrix}$$

Again, nodewise enforcement of these conditions can be obtained by replacing shape functions with Dirac delta functions.

For the regularized problem some additional artificial terms (fluxes) occur on the solid wall due to second-order terms and explicit artificial dissipation. These fluxes are forced to be zero by means of natural boundary conditions, in the same manner as the supersonic outflow.

No-Flow

The basic condition of the no-flow boundary is that the normal velocity is zero or, equivalently, that the normal momentum is zero,

$$m_i n_i = 0$$

In the incremental form this becomes

$$\Delta m_i n_i = -m_i^n n_i$$

This condition is easily enforced by a penalty function, with the addition of the term

$$\frac{1}{\epsilon}\int_{\partial\Omega_W} (\Delta m_i n_i + m_i^n n_i)\, V_{(1+i)} ds \tag{2.71}$$

in the variational formulation, where $V_{(1+i)}$ is the component of a test function corresponding to momentum m_i. The resulting stiffness matrices and right-hand sides are of the standard form (2.70), with kernels (in two dimensions):

$$\boldsymbol{k} = \frac{1}{\epsilon}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & n_1 n_1 & n_1 n_2 & 0 \\ 0 & n_2 n_1 & n_2 n_2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{r} = -\frac{1}{\epsilon}(m_i n_i) \begin{bmatrix} 0 \\ n_1 \\ n_2 \\ 0 \end{bmatrix}$$

For viscous flow, the additional conditions on the no–flow boundary are the two natural boundary conditions:

$$\begin{aligned} \sigma_{ns} &= 0 \\ q_n &= 0 \end{aligned}$$

where σ_{ns} is the skin friction on the boundary and q_n is the normal heat flux. The application of these natural boundary conditions follows the procedure discussed in preceding subsections. Similarly, as on the solid wall, all artificial fluxes are forced to be zero on the no–flow boundary.

2.3.2 Boundary Conditions for the Two–Step Method

This section presents a brief overview of the boundary conditions for the two–step algorithm outlined in Section 2.2.2. In general, the boundary condition can be constructed separately for the convection (Euler) step and the diffusion (viscous) step. This holds provided that they guarantee stability of those steps and possess appropriate asymptotic properties as the viscosity constants approach zero ($Re_L \rightarrow \infty$). With this in mind, we summarize the different classes of boundary conditions appropriate for the Euler step, momentum step, and energy step.

Euler Step

The boundary conditions for the Euler step in general follow directly from the one–step algorithm with the following special conditions.

1. Contributions of viscous fluxes to boundary terms in variational equation (2.41) are omitted.

2. The essential boundary condition on a solid wall is limited to enforcing a zero normal component of the momentum vector. It is accomplished by means of a penalty method, i.e., by adding the following contributions to the stiffness matrix:

$$\frac{1}{\varepsilon}\int_{\partial\Omega}(U_2 n_x + U_3 n_y)(\delta U_2 n_x + \delta U_3 n_y)\, dS$$

   where ε is a small penalty parameter.

Boundary Conditions for the Momentum Step

Boundary conditions for the momentum and energy steps were constructed such that the viscous term in expression (2.59) results in a positive production of entropy and the resulting boundary terms in boundary value problem (2.57) and (2.58) make these problems well posed. These boundary conditions can be listed as follows:

Case 1 **Open Boundary — Supersonic Inflow**

Full Dirichlet boundary conditions are prescribed,

$$m_i^{n+1} = m_i^{in} \tag{2.72}$$

where m_i^{in} are the momentum components of the same inflow vector as used in the supersonic inflow boundary conditions for the Euler step.

Case 2 **Open boundary — Subsonic Inflow**

The same Dirichlet boundary conditions are used, but with the inflow vector replaced with the solution from the Euler step, i.e.,

$$m_i^{n+1} = m_i^n \tag{2.73}$$

Case 3 **Open Boundary — Subsonic Outflow**

Mixed boundary conditions are used,

$$\begin{aligned} m_n^{n+1} &= m_n^n \\ \tau_s^{n+1} &= \tau_s^n \end{aligned} \tag{2.74}$$

where m_n is the normal component of the momentum,

$$m_n = m_1 n_1 + m_2 n_2 \tag{2.75}$$

and τ_s is the tangential viscous stress vector component,

$$\tau_s = (\tau_{22} - \tau_{11}) n_1 n_2 + \tau_{12}(n_1^2 - n_2^2) \tag{2.76}$$

Case 4 **Open Boundary — Supersonic Outflow**

Full Neuman boundary conditions are applied,

$$\tau_{ij}^{n+1} n_j = \tau_{ij}^n n_j \tag{2.77}$$

Case 5 **Solid Wall Boundary Conditions**

Full Dirichlet boundary conditions are used,

$$m_i^{n+1} = 0 \tag{2.78}$$

Case 6 **Symmetry Boundary Conditions (of the first kind)**

Mixed boundary conditions are applied,

$$\begin{aligned} m_n^{n+1} &= 0 \\ \frac{\partial m_s^{n+1}}{\partial n} &= 0 \end{aligned} \tag{2.79}$$

where m_n and m_s are the normal and tangential components of the momentum vector.

Case 7 **Symmetry Boundary Conditions (of the second kind)**

Full Neuman boundary conditions are applied:

$$\frac{\partial m_i^{n+1}}{\partial n} = 0 \tag{2.80}$$

As in the case of Euler equations, substituting (2.80) into the boundary integral in (2.57), results in some non-zero terms which must be included in the stiffness matrix calculations.

In the finite element code, all of the essential boundary conditions have been implemented using the penalty approach, i.e., replacing the full Dirichlet boundary conditions with

$$m_i^{n+1} + \varepsilon\beta\Delta t \sum_{j=1}^{2} \tau_{ij}^{n+1} n_j = m_i^{in} \tag{2.81}$$

and the first of (2.74) conditions with

$$m_n^{n+1} + \varepsilon\beta\Delta t \tau_n^{n+1} = m_n^n \tag{2.82}$$

where τ_n^{n+1} is the normal viscous stress

$$\tau_n = \sum_{i,j=1}^{2} \tau_{ij} n_i n_j \tag{2.83}$$

Boundary Conditions for the Energy Step

Case 1 Temperature (energy) Prescribed

A single Dirichlet boundary condition is applied

$$e^{n+1} = \overline{e} \tag{2.84}$$

The choice of $\overline{e}$ will vary with the particular kind of boundary:

- for supersonic inflow $\overline{e}$ corresponds to the inflow vector.
- for subsonic inflow $\overline{e}$ is evaluated using the solution from the previous step.
- for a solid wall with temperature prescribed $\overline{e}$ corresponds to the prescribed temperature on the wall and the density ρ (unchanged in the viscous step).

Note that the two-step method eliminates the contradictions resulting from the discussion of the solid wall boundary conditions with temperature prescribed, as the density remains unchanged (see [10]).

Case 2 Heat Flux Prescribed

A single Neumann boundary condition is applied:

$$\kappa \frac{\partial T^{n+1}}{\partial n} = \overline{q} \tag{2.85}$$

The heat flux $\overline{q}$ is calculated in the following way:

- for subsonic and supersonic outflow, $\overline{q}$ is evaluated using the solution from the previous step
- for an adiabatic wall and symmetry boundary conditions of both kinds, $\overline{q}$ is assumed to be zero.

2.4 Artificial Viscosity

In order to suppress spurious oscillation of the solution an artificial dissipation is introduced as an additional flux in the Navier–Stokes equations in the form

$$\boldsymbol{U}_{,t} + \boldsymbol{F}^{C}_{i,i} = \boldsymbol{F}^{V}_{i,i} + \boldsymbol{F}^{A}_{i,i} \tag{2.86}$$

where $\boldsymbol{F}^{V}_{i}$ denotes the artificial dissipation flux with corresponding Jacobians defined as

$$\boldsymbol{P}^{A}_{i} = \frac{\partial \boldsymbol{F}^{A}_{i}}{\partial \boldsymbol{U}} \qquad \boldsymbol{R}^{A}_{ij} = \frac{\partial \boldsymbol{F}^{A}_{i}}{\partial \boldsymbol{U}_{,j}}$$

The advantage of this approach is that the artificial dissipation can be treated using exactly the same formulation and procedures as for the natural viscosity. In the one–step algorithm, for the sake of generality, a fourth implicitness parameter γ is introduced for the terms associated with the artificial dissipation. In the calculation of the stiffness matrices, right–hand sides and boundary terms, the same formulas are used as for the natural viscosity. Similarly, for the two–step algorithm, the artificial viscosity is implemented as an additional "viscous" flux when solving the equation of the Euler equation.

In this work, two commonly used forms of artificial viscosity have been studied and implemented as follows:

- the classical Lapidus viscosity [20]

$$\boldsymbol{F}_i^A = k_{ii}\boldsymbol{U}_{,i} \tag{2.87}$$

with

$$k_{ii} = c_k h^2 \left|u_{i,i}\right| \tag{2.88}$$

The Jacobians $\boldsymbol{P}^A$ and $\boldsymbol{R}^A$ can be defined by a straightforward differentiation of (2.86).

- the generalized Lapidus viscosity due to Löhner, et al. [23]

$$\boldsymbol{F}^A = \boldsymbol{\ell} k \frac{\partial \boldsymbol{U}}{\partial \boldsymbol{\ell}} \qquad \text{or} \qquad \boldsymbol{F}_i^A = k l_i l_j \boldsymbol{U}_{,j} \tag{2.89}$$

with

$$\begin{aligned} k &= c_k h^2 \left(\boldsymbol{\ell} \cdot \nabla(\boldsymbol{u} \cdot \boldsymbol{\ell})\right) \\ \boldsymbol{\ell} &= \nabla|\boldsymbol{u}| / \left|\nabla|\boldsymbol{u}|\right| \end{aligned} \tag{2.90}$$

where h is the element size, c_k is an arbitrary coefficient controlling the amount of dissipation, k is a solution dependent set of coefficients, $\boldsymbol{\ell}$ is a unit vector parallel to $\nabla|\boldsymbol{u}|$, and $\boldsymbol{u}$ is the velocity vector. The tensor product $l_i l_j$ ensures that the artificial viscosity acts in the direction normal to the shocks. The Jacobians $\boldsymbol{P}^A$ and $\boldsymbol{R}^A$ can be defined by differentiation of (2.88). For simplicity, dependence of k and $\boldsymbol{\ell}$ on the solution is disregarded, in the definition of jacobians so that

$$\boldsymbol{P}_i^A = 0, \qquad \boldsymbol{R}_{ij}^A = k l_i l_j \boldsymbol{I},$$

where $\boldsymbol{I}$ is the identity matrix.

The drawback of the above formulations is that for elements with high aspect ratios the element size h is not a clearly defined quantity. The wrong choice of h may cause oscillations

if h is too small or considerable smearing of shocks if h is too large. We have developed a modification of the generalized Lapidus viscosity (2.88) such that it uses more precise information about the geometry of the element than just h but still preserves its original form for square elements. In terms of the contribution to the element stiffness matrix, the generalized Lapidus viscosity can be written in a slightly different form as

$$c_k \Delta t \, h^2 \int_{\Omega_K} \boldsymbol{V}_{,i} \, k_{ij} \, \boldsymbol{U}_{,j} d\Omega \tag{2.91}$$

where $k_{ij} = l_i l_j k$ and k is a solution dependent scalar

$$k = \left| \frac{\partial(\boldsymbol{u} \cdot \boldsymbol{\ell})}{\partial \boldsymbol{\ell}} \right| \tag{2.92}$$

The basic idea of constructing the modified artificial vsicosity is to perform the calculation of (2.90) on the master element, which has a fixed size ($h = 2$), then map it back to the physical element domain. As a result of this procedure the modified artificial viscosity is of the form

$$c_k \Delta t \, h^2 \int_{\Omega_K} \boldsymbol{V}_{,i} \, \tilde{k}_{ij} \, \boldsymbol{U}_{,j} d\Omega \tag{2.93}$$

where $\tilde{k}_{ij}$ is given as

$$\tilde{k}_{ij} = \frac{\partial x_i}{\partial \xi_p} \, k_{pq} \, \frac{\partial x_j}{\partial \xi_q} \tag{2.94}$$

and k_{pq} is of the same form as in (2.89) except that the unit vector $\boldsymbol{\ell}$ is taken as

$$\tilde{\boldsymbol{\ell}} = \nabla_\xi |\boldsymbol{v}| \, / \, |\nabla_\xi |\boldsymbol{v}|| \tag{2.95}$$

to make the viscosity work in the direction perpendicular to the shocks on the master element. ∇_ξ denotes the gradient calculated on the master element coordinate. It can be shown that for square element this modified artificial viscosity does coinside with the original expression (2.90). We have applied this modified viscosity in several test problems where element with high aspect ratios are used, and have found it to be more effective than the artificial viscosities defined by equations (2.88) and (2.90).

3 An h-p Finite Element Method

The aim of adaptive methods in computational fluid dynamics is to optimize the computational process: to obtain the best results for the least effort. The cost functional in this optimization process is the numerical error measured in some appropriate norm, both the global error over the entire computational domain and the local error over each gridcell. The central parameters are the conventional mesh parameters that govern local accuracy: the mesh size h, the order p (e.g., the spectral order) of the local approximation, and the location of gridpoints.

In the h-p FEM one can control both the local mesh size and spectral order of approximation simultaneously. Such a flexibility allows one to distribute degrees of freedom in an optimal way: a large density of degrees of freedom can be used in computational regions with very irregular behavior of the solution while a relatively rough approximation is used in subdomains where the solution is smooth. This suggests that the h-p method may use an optimal number of degrees of freedom to achieve a prescribed accuracy. In addition, recent work in the area of approximation theory [2,3] suggests that an extra gain in accuracy can be obtained if the enrichment of the mesh is performed in two combined ways: first by reducing the size of elements h and second by increasing their spectral order p. The problem of how to combine these two kinds of refinements so that the improvement in accuracy is the best possible is a very complex issue. In general, its strict mathematical solution is not known, however, there exists heuristical knowledge on the use of h-p FEMs for many classes of problems.

In practice the reduction of the mesh size h can be achieved in two ways: by subdividing elements into smaller sons, or by so–called remeshing, i.e., generating a completely new mesh with a given distribution of h. Our implementation of the h-p FE method uses the first approach. We break two–dimensional quadrilateral elements into 4 element sons and three–dimensional hexagonal elements into 8 sons.

The success of such a complex adaptive scheme depends upon several properties of the adaptive process: the data structure, the adaptive strategy, the techniques for *a posteriori* error estimation, and the flow solver. The potential payoff of a successful h-p adaptive strategy is substantial: exponential rates of convergence may be attained, meaning that complex flow features can be resolved using orders-of-magnitude fewer unknowns than required by conventional methods.

3.1 A Variational Formulation

Let us first specify the class of problems to be solved with an h-p FEM. In this development, we follow the detailed presentation in our papers [9, 11, 25].

Let Ω be an open bounded domain in $\mathbb{R}^n$, $n = 2, 3$ with a sufficiently regular boundary $\partial\Omega$. In what follows, we shall restrict ourselves to a class of problems that can be formulated in the following abstract form:

$$\left.\begin{array}{l} \text{Find } \boldsymbol{u} \in \boldsymbol{X} \text{ such that} \\ \\ B(\boldsymbol{u}, \boldsymbol{v}) = L(\boldsymbol{v}) \quad \forall \boldsymbol{v} \in \boldsymbol{X} \end{array}\right\} \tag{3.1}$$

Here:

$$\boldsymbol{X} = X \times X \cdots \times X (m \text{ times}) \tag{3.2}$$

where X a subspace of $H^1(\Omega)$, the Sobolev space of first order, $B(\cdot, \cdot)$ is a bilinear form on $\boldsymbol{X} \times \boldsymbol{X}$ of the following form

$$B(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i,j=1}^{m} B_{ij}(u_i, v_j) \tag{3.3}$$

where $B_{ij}(\cdot, \cdot)$ are bilinear forms of scalar-valued arguments of the type

$$\begin{aligned} & B_{ij}(u, v) \\ & = \int_\Omega \left\{ \sum_{k,l=1}^{n} a_{ij}^{kl} \frac{\partial u}{\partial x_k} \frac{\partial v}{\partial x_l} + \sum_{k=1}^{n} b_{ij}^{k} \frac{\partial u}{\partial x_k} v + c_{ij} uv \right\} dx \\ & + \int_{\partial\Omega} d_{ij} uv \, ds \end{aligned} \tag{3.4}$$

and

$$L(\cdot) \text{ is a linear form on } \boldsymbol{X} \tag{3.5}$$

of the form

$$L(\boldsymbol{v}) = \sum_{j=1}^{m} L_j(v_j) \tag{3.6}$$

with the linear forms $L_j(\cdot)$ acting on the scalar-valued functions

$$L_j(v) = \int_\Omega \left\{ f_j v + \sum_{k=1}^{n} g_j^k \frac{\partial v}{\partial x_k} \right\} dx + \int_{\partial\Omega} h_j v ds \tag{3.7}$$

In the above formulas, a_{ij}^{kl}, b_{ij}^{k}, c_{ij}, f_i, g_j^k, d_{ij}, h_j are sufficiently regular functions defined on Ω and on $\partial\Omega$, respectively.

Numerous examples fall into the category of problems described by the abstract formulation (3.1). To mention a few: linear elliptic problems (both single equations and systems), linear problems resulting from a one–step approximation in time for evolution problems, linear steps of a nonlinear problem solution, etc.

In this formulation, both the solution $\boldsymbol{u}$ and the test functions $\boldsymbol{v}$ are members of the same space $\boldsymbol{X}$. Non–homogeneous essential boundary conditions are handled by means of a standard penalty approach.

3.2 Finite Element Approximation

We assume that the domain Ω can be represented as a union of finite elements K_e, $e = 1,\ldots,M$. More precisely

$$\overline{\Omega} = \bigcup_{e=1}^{M} \overline{K}_e$$

and

$$\text{int } K_e \cap \text{ int } K_f = \emptyset \quad \text{for } e \neq f$$

Each of the elements K has a corresponding finite dimensional space of shape functions, denoted $X_h(K)$, for instance the space of polynomials of order p. The global finite element space X_h consists of functions which, when restricted to element K, belong to the local space of shape functions $X_h(K)$. Thus the global approximation is constructed by patching together the local shape functions in the usual way.

We shall adopt the fundamental requirement that the global approximation must be *continuous*. As we will see, this requirement leads to the notion of constrained approximations. Formally, the continuity assumption guarantees that the finite element space X_h is a subspace of $H^1(\Omega)$ and, with some additional assumptions if necessary, also a subspace of X. The approximate problem is easily obtained from (3.1) by substituting for $\boldsymbol{u}$ and $\boldsymbol{v}$ their approximations $\boldsymbol{u}_h$ and $\boldsymbol{v}_h$:

$$\left.\begin{array}{ll} \text{Find } \boldsymbol{u}_h \in \boldsymbol{X}_h & \text{such that} \\ B_h(\boldsymbol{u}_h, \boldsymbol{v}_h) = L_h(\boldsymbol{v}_h) & \forall \boldsymbol{v}_h \in \boldsymbol{X}_h \end{array}\right\} \tag{3.8}$$

Here

$$\boldsymbol{X}_h = X_h \times \ldots \times X_h (m \text{ times}) \tag{3.9}$$

which indicates that the *same* approximation has been applied to every component of $\boldsymbol{u}$. $B_h(\cdot,\cdot)$ and $L_h(\cdot)$ denote approximations to the original bilinear and linear forms resulting from numerical integration.

3.3 Adaptivity

A flowchart of a typical Adaptive Finite Element Method (AFEM) is shown in Fig. 3.1. The method consists of first generating an initial mesh and solving for the corresponding FEM approximate solution. Next, the error is estimated in some way and based on this (usually crude) approximation, one *adapts* the mesh, i.e., adds new degrees of freedom. The approximate problem for the new mesh is solved *again* and the whole procedure continues until certain error tolerances are met. Obviously, such a procedure requires an estimate of the error over each element and a strategy to reduce the error by proper changes in the mesh parameters, h and p.

In our adaptive FEM the new degrees of freedom can be added in two different ways: elements may be locally refined or their spaces of shape functions may be enriched by incorporating new shape functions. As noted earlier, in the case of polynomials, this is done by increasing locally the degree of polynomials used to construct the shape functions, the first case being an h-refinement, and the second case a p-refinement. A combination of both is an (adaptive) h-p FEM. We remark that the process of increasing the local polynomial degrees for a fixed mesh size is mathematically akin to increasing the spectral order of the approximation and that, therefore, we also refer to h-p methods as "adaptive spectral–element" or "h-spectral" methods.

3.4 Regular and Irregular Meshes

As the result of local h-refinements, *irregular meshes* are introduced. Recall (see [26]) that a *node is called regular* if it constitutes a vertex for each of the neighboring elements; otherwise it is *irregular*. If all nodes in a mesh are regular, then the mesh itself is said to be *regular*. In the context of two–dimensional meshes, the maximum number of irregular nodes on an element side is referred to as the *index of irregularity*. Meshes with an index of irregularity equal one are called *1-irregular meshes*. The notion can be easily generalized to the three–dimensional case. (See [7] and literature cited therein for additional references.)

In the present work, we accept only 1-irregular meshes. In the two–dimensional context, this translates into the requirement that a "large" neighbor of an element may have no more than two "small" neighbors on a side; in the three–dimensional case, the number of neighbors sharing a side may go up to four, while the number of neighbors sharing an edge can be no more than two. This is frequently called the *"two-to-one"* rule (cf. [7]). Examples of regular and irregular meshes are shown in Fig. 3.2. There are several practical and theoretical reasons to accept only 1-irregular meshes, especially in the context of h-p methods. For a detailed argument, we refer to [4].

Our restriction to 1-irregular meshes imposes a simple restriction on the way any h-

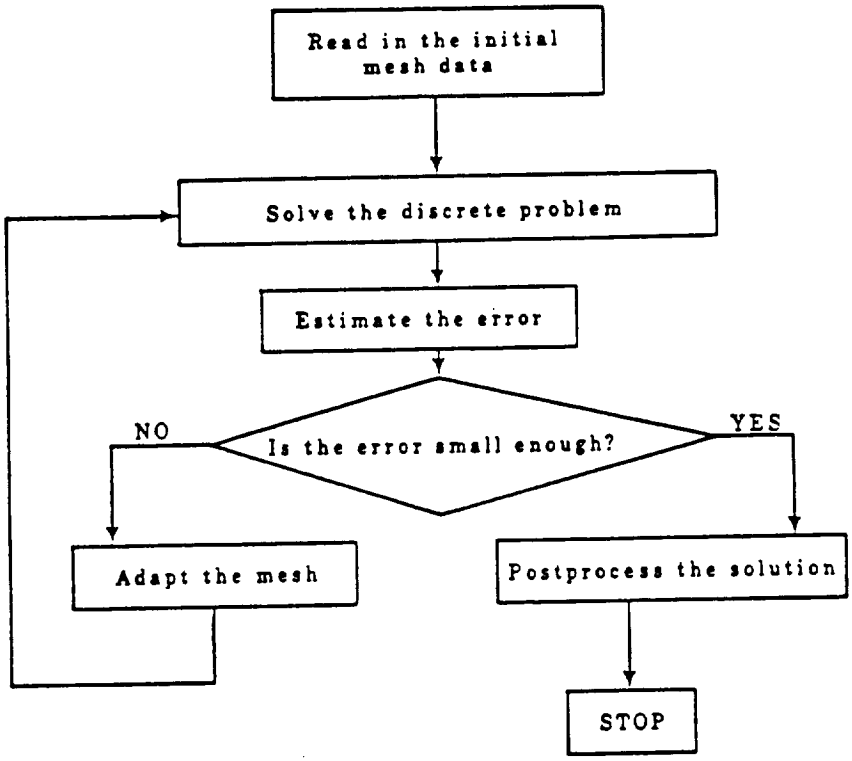


Figure 3.1: Typical flowchart of an adaptive method.

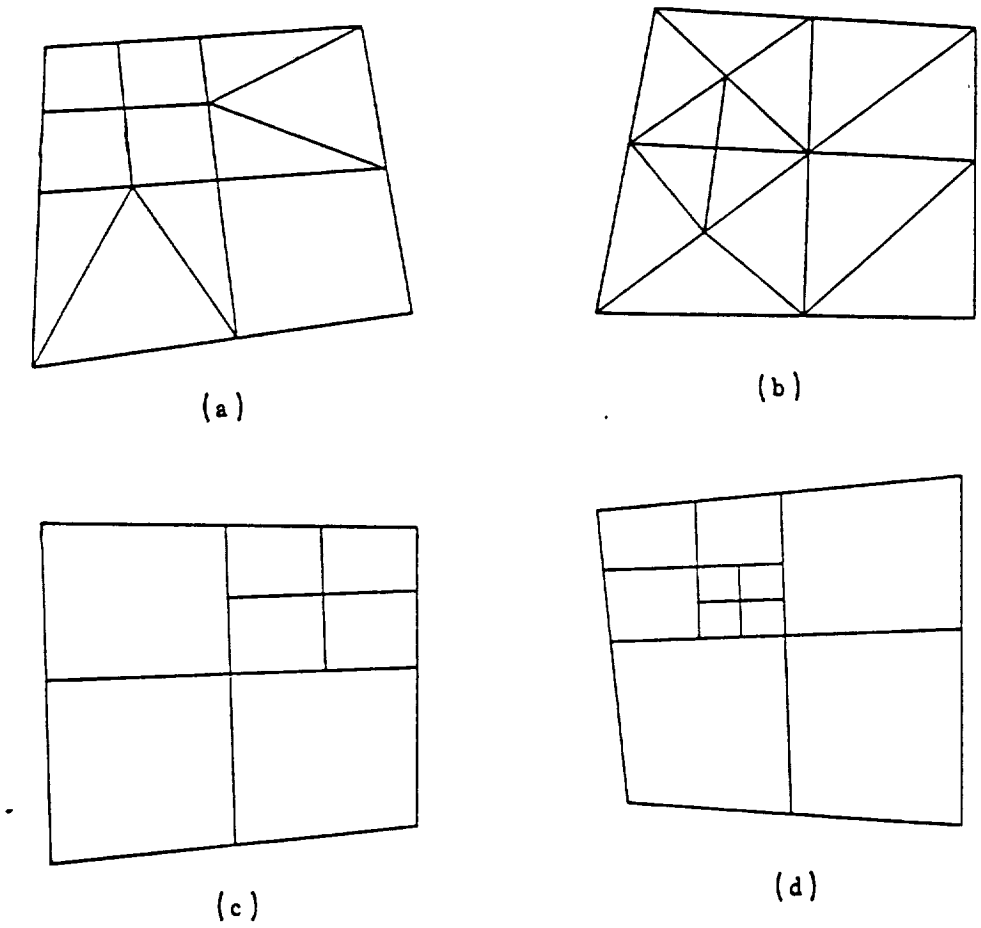


Figure 3.2: Examples of regular and irregular meshes: (a) and (b) — regular mesh; (c) 1-irregular mesh (index of irregularity = 1); (d) 2-irregular mesh.

refinement can proceed: before an element is refined, a check for "larger" neighbors must be made. If any such neighbors exist, they must be refined first and only then can the element in question be refined.

3.5 Basic Assumptions

As indicated in the previous sections, the construction of an h-p FEM is based on the following assumptions:

- only 1-irregular meshes are accepted for all h-refinements;
- the local order of approximations may differ in each element;
- the approximation must be continuous.

3.6 Definition of an Element

The classical definition of an element is a triple

$$\{K, X, \varphi_i, i = 1, \ldots, N\}$$

where K is the domain of the element (a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$), X is an N-dimensional space of shape functions:

$$X \ni \varphi_i \colon K \to \mathbb{R},\ i = 1, \ldots, N$$

and φ_i, $i = 1, \ldots, N$ is a set of degrees of freedom, i.e., a set of linearly independent linear functionals on X.

The element base shape functions χ_i are understood as a dual basis to φ_i:

$$\chi_i \in X \text{ such that}$$
$$\langle \varphi_i, \chi_j \rangle = \delta_{ij},\ i, j = 1, \ldots, N$$

Following this classical construction we define a two–dimensional quadrilateral element as follows. In the first step we introduce a two–dimensional master element

$$\left\{\widehat{K}, \widehat{X}, \widehat{\varphi}_i, i = 1, \ldots, N\right\}$$

The domain $\widehat{K}$ is a unit square, $\widehat{K} = [-1, 1]^2$. The space of shape functions $\widehat{X}$ is a subset of $Q^p(\widehat{K})$, i.e., polynomials of the order p in each variable. We define this subset in such a way that $\widehat{X}$ has the following properties:

i) Each function in $\widehat{X}$ can be associated with one of nine nodes of the element: four corners, four midside nodes and the centroid; the maximum order of a shape function associated with a given node is viewed as the order of this node.

ii) The base shape functions constituting $\widehat{X}$ will be so-called *hierarchical shape functions*, which means that enriching the element from the order p to $p+1$ consists of *adding some higher order functions to* $\widehat{X}$ *without modifying the functions already belonging to* $\widehat{X}$.

Space $\widehat{X}$ with these properties is constructed as follows: First we introduce one–dimensional hierarchical shape functions on $[-1,-1]$:

$$\chi_0 = \frac{1}{2}(1-x) \tag{3.10}$$

$$\chi_1 = \frac{1}{2}(1+x) \tag{3.11}$$

$$\chi_2 = x^2 - 1 \tag{3.12}$$

$$\chi_3 = x^3 - x \tag{3.13}$$

$$\chi_4 = x^4 - 1 \tag{3.14}$$

$$\ldots \tag{3.15}$$

(Figure 3.3) The corresponding degrees of freedom can be associated with the two endpoints -1 and 1 and the midpoint 0:

$$\langle \varphi_0, u \rangle = u(-1) \tag{3.16}$$

$$\langle \varphi_1, u \rangle = u(1) \tag{3.17}$$

$$\langle \varphi_i, u \rangle = \lambda_i^{-1} \frac{d^i u}{dx^i}, \; i = 2,3,\ldots \tag{3.18}$$

where λ_i are scaling factors.

Note that the linear functions χ_0, χ_1 assume values $0,1$ at the endpoints while all the higher order functions vanish at ± 1.

Then for a two–dimensional element we associate the following functions with the subsequent nodes:

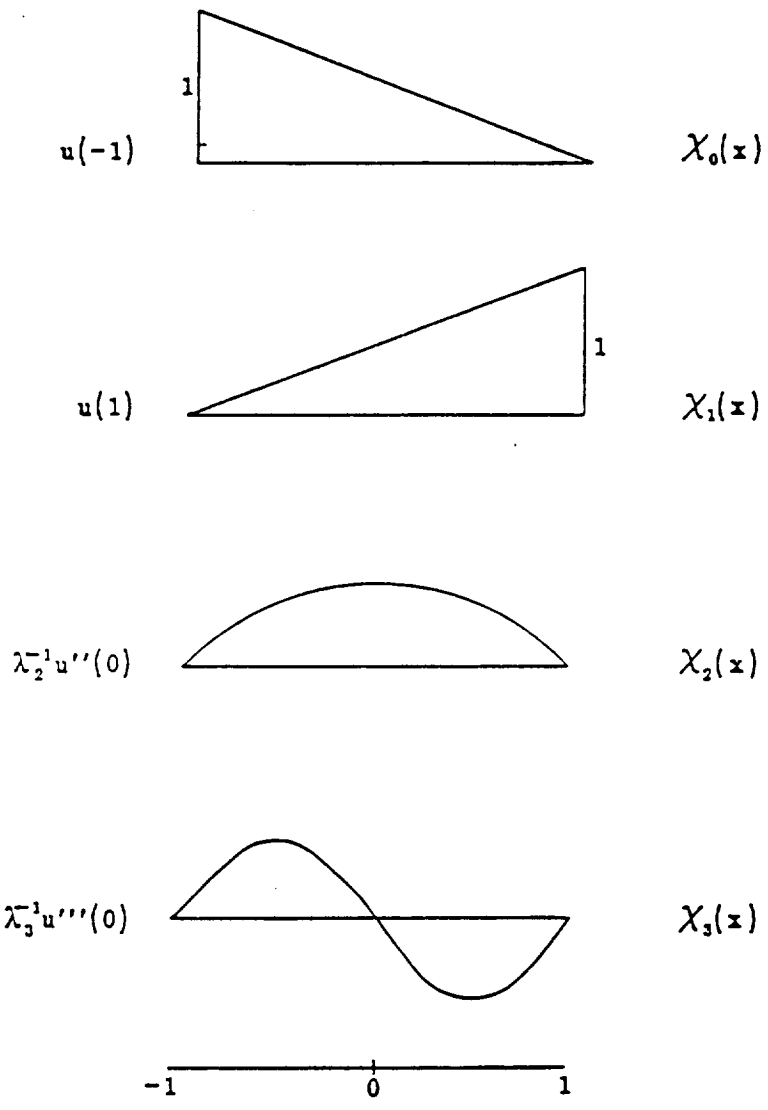


Figure 3.3: One-dimensional hierarchical master elements. Degrees of freedom and corresponding shape functions.

i) The bilinear functions

$$\chi_0(x)\chi_0(y),\ \chi_0(x)\chi_1(y),\ \chi_1(x)\chi_0(y),\ \chi_1(x)\chi_1(y)$$

with corner nodes $x, y = \pm 1$.

ii) The functions linear in one direction and higher order in the other, with midside nodes:

$$\begin{aligned}
&\chi_2(x)\chi_0(y),\ldots,\chi_{p_1}(x)\chi_0(y),\\
&\chi_2(x)\chi_1(y),\ldots,\chi_{p_3}(x)\chi_1(y)\\
&\chi_0(x)\chi_2(y),\ldots,\chi_0(x)\chi_{p_4}(y)\\
&\chi_1(x)\chi_2(y),\ldots,\chi_1(x)\chi_{p_2}(y)
\end{aligned}$$

- The functions at least quadratic in both directions, with the centroid:

$$\chi_i(x)\chi_j(y)\,,\ i,j = 2,\ldots,p_5$$

where $p_1,\ldots,p_4$ denote degrees of approximation of midside nodes, p_5 the degree of a centroid node (see Fig. 3.4).

The above set of shape functions is dual to degrees of freedom which are tensor products of one-dimensional degrees of freedom given by (3.18). These degrees of freedom can be listed as follows:

- function values at four vertices:

$$u(-1,-1), u(1,-1), u(1,1), u(-1,1) \tag{3.19}$$

- tangential derivatives (up to a multiplicative constant) up to p-th order associated with the midpoints of the four edges:

$$\begin{aligned}
&\lambda_k^{-1}\frac{\partial^k u}{\partial x^k}(0,-1) \quad k = 2,\ldots,p_1\\
&\lambda_k^{-1}\frac{\partial^k u}{\partial y^k}(1,0) \quad k = 2,\ldots,p_2\\
&\lambda_k^{-1}\frac{\partial^k u}{\partial x^k}(0,1) \quad k = 2,\ldots,p_3\\
&\lambda_k^{-1}\frac{\partial^k u}{\partial y^k}(-1,0) \quad k = 2,\ldots,p_4
\end{aligned} \tag{3.20}$$

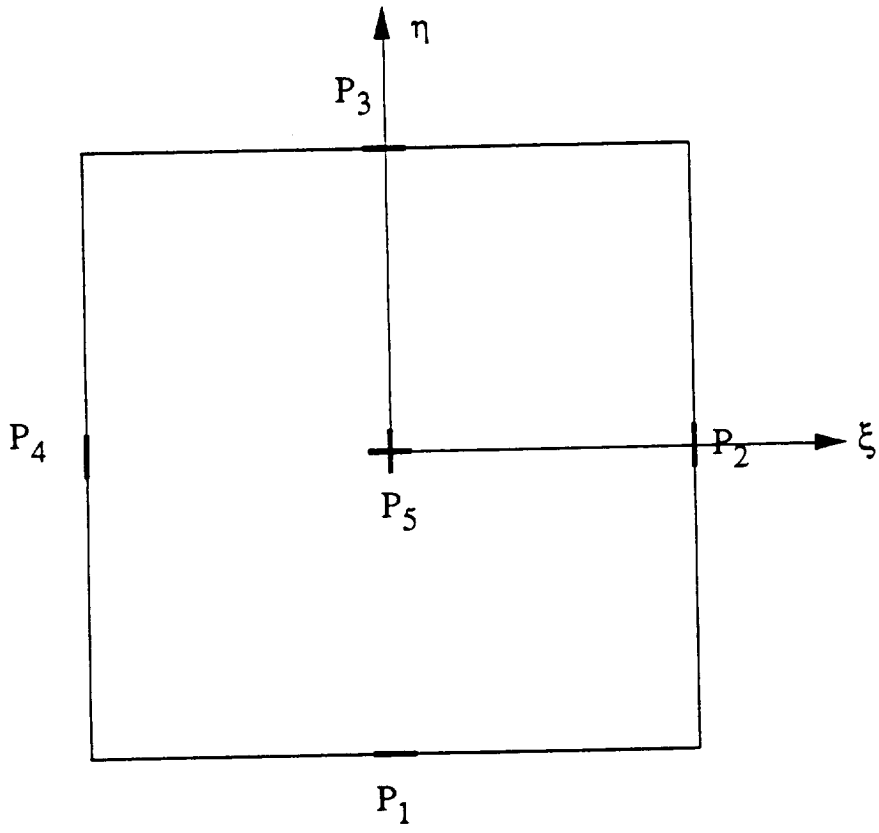


Figure 3.4: Association of orders of approximation with nodes.

- mixed order derivatives associated with the central node

$$\lambda_k^{-1}\lambda_\ell^{-1}\frac{\partial^{k+l}u}{\partial x^k \partial y^\ell}(0,0) \quad k,l = 2,\ldots,p_5 \tag{3.21}$$

Having constructed the master element we define a subparametric element K. It is obtained by mapping of the master element $\widehat{K}$ into an actual computational domain:

$$K = T\left(\widehat{K}\right)$$

The mapping T is defined as

$$T(\hat{x},\hat{y}) = \sum_{i=1}^{9} \boldsymbol{a}_i N_i(\hat{x},\hat{y})$$

where $N_i, i = 1,\ldots,9$ are the regular (Lagrangian) shape functions for the 9-node biquadratic element, $\boldsymbol{a}_i$ are the desired positions of nodes of K in the computational domain. The space of shape functions of K is taken as a space of compositions of T and $\hat{\psi}_i \in \widehat{X}$:

$$X = \left\{\psi_i : K \to \mathbb{R} \,|\, \psi_i = \hat{\psi}_i T^{-1}\right\}$$

The degrees of freedom of K are defined using $\hat{\varphi}_i$'s:

$$\langle \varphi_i, \psi_j \rangle \stackrel{\text{def}}{=} \left\langle \hat{\varphi}_i, \hat{\psi}_j \right\rangle$$
$$\text{with } \hat{\psi}_j = \psi_j T$$

The definition of a two–dimensional element given above can easily be generalized to the three–dimensional case. For completeness, we outline the major steps of this construction:

We introduce cube subparametric elements: actual elements are images of a cube master element $\widehat{K} = [-1,1]^3$ under the mapping $T: \widehat{K} \to K \subset \mathbb{R}^3$

$$x^i = \sum_{j=1}^{27} x_j^i N_j(\boldsymbol{\xi}), \tag{3.22}$$

where N_i are second order Lagrangian polynomials associated with 27 element nodes: 8 corners, 12 midpoints of edges, 6 centers of walls and one central node.

We equip a master element with the space $X_h(\widehat{K})$ of p-th order hierarchical shape functions defined as a triple tensor product of a set of one–dimensional hierarchical shape functions $\chi_i(\cdot)$ on an interval $[-1,1]$. Actual shape functions of K, $X_h(K)$ are as usual compositions of the mapping T^{-1} and shape functions on $\widehat{K}$:

$$X_h(\widehat{K}) = \left\{u = \hat{u}T^{-1} \mid \hat{u} \in X_h(\widehat{K})\right\}. \tag{3.23}$$

Degrees of freedom $\hat{\varphi}$ associated with the hierarchical base shape functions on a master element are defined as follows:

- function values at 8 corners:

$$u(\pm 1, \pm 1, \pm 1),$$

- Tangential derivatives (up to a multiplicative constants) up to p-th order associated with the midpoints of edges:

$$\lambda_k^{-1} \frac{\partial^k u}{\partial s^k}, \quad k = 2, \ldots, p,$$

  where s is a coordinate parallel to the edge,

- mixed order derivatives associated with centers of walls:

$$\lambda_k^{-1} \lambda_l^{-1} \frac{\partial^{k+l} u}{\partial s^k \partial r^\ell}$$

  where s, r is a pair of coordinates parallel to the wall,

- mixed order derivatives associated with a centroid:

$$\lambda_k^{-1} \lambda_l^{-1} \lambda_m^{-1} \frac{\partial^{k+l+m} u}{\partial x^k \partial y^l \partial z^m}.$$

Degrees of freedom of the actual element K are defined as

$$\langle \varphi, u \rangle \stackrel{\text{def}}{=} \langle \hat{\varphi}, \hat{u} \rangle$$

3.7 Continuity for Regular Meshes

One of the fundamental advantages of using the hierarchical shape functions is the ease with which they allow one to construct a continuous approximation with locally variable order of approximation. A typical situation is illustrated in Fig. 3.5. If elements K_1 and K_2 are to support polynomials of degree, say, one and three, respectively, then there are at least two ways to enforce continuity across the interelement boundary. One way is to add two extra shape functions of second and third order corresponding to the middle node A of element K_1. Alternatively, the same two shape functions may be deleted from element K_2. In all these cases, a *common order of approximation* along the interelement boundary can be enforced by simply adding or deleting the respective shape functions from the neighboring elements. While any of these choices can be made, the results described here employ the "maximum rule" in which the higher-order approximation dominates lower orders. Thus, if an element is p-refined, i.e., a higher order approximation of degree $\bar{p}$ is introduced, the neighbors of lower order are enriched by the addition of extra shape functions of degree $\bar{p}$ necessary to guarantee continuity of the approximation.

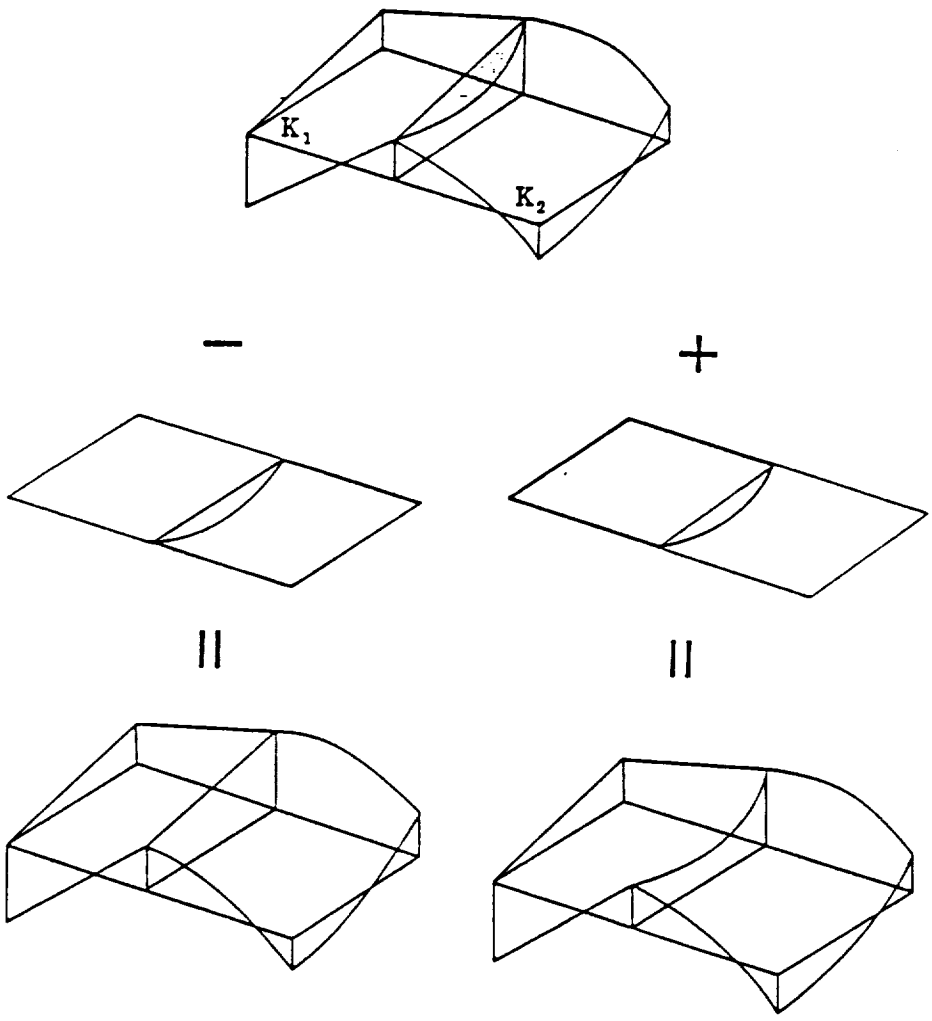


Figure 3.5: Continuity by hierarchical shape functions. (cf [9])

3.8 Continuity for 1-Irregular Meshes. Constrained Approximation

Continuity of the approximation on 1-irregular meshes is a more complicated issue. It leads to the notion of a constrained approximation that we introduce in this section.

Consider two adjacent square elements in a 1-irregular mesh, one having an irregular ("hanging") corner node on the side of the other (Fig. 3.6). The first condition that must be satisfied to make the approximation continuous across the common boundary Γ_f is that the spaces of shape functions of elements K_e and K_f, if restricted to Γ_f, be identical:

$$X_h(K_e)|_{\Gamma_f} = X_h(K_f)|_{\Gamma_f} \tag{3.24}$$

This means, of course, that the orders of approximation for nodes 1 and 2 (Fig. 3.6) must be equal. Condition (3.17) is exactly the same as that for regular meshes considered before.

Denote by $\psi_{i,K_e}(\boldsymbol{x})$, $\psi_{i,K_f}(\boldsymbol{x})$ the base shape functions of elements K_e and K_f which do not identically vanish along Γ_f.

Assume that the common order of nodes 1 and 2 is p. Since spaces $X_h(K_e)|_{\Gamma_f}$ and $X_h(K_f)|_{\Gamma_f}$ are identical there must exist a unique linear invertible relation between functions constituting the basis of these spaces:

$$\psi_{i,K_e}(\boldsymbol{x})|_{\Gamma_f} = \sum_{j=0}^{p} {}^dR_{ij}\psi_{j,K_f}(\boldsymbol{x})|_{\Gamma_f},\ i = 0,\ldots,p \tag{3.25}$$

In this formula, the functions involved are assigned indices from 0 to p corresponding to orders of the functions. Coefficients ${}^dR_{ij}$ are calculated in one of the next sections. The superscript d distinguishes between the two generic situations: K_f may be attached to the left or the right part of K_e. Formula (3.25) implies that the degrees of freedom of K_e and K_f corresponding to functions involved in (3.25) are also related by a similar equation: take any *continuous* function $u_h(\boldsymbol{x})$ such that $u_h|_{K_e} \in X_h(K_e)$, $u_h|_{K_f} \in X_h(K_f)$. Then its restriction to Γ_f can be written as:

$$u_h|_{\Gamma_f} = \sum_{i=0}^{p} U_i\psi_{i,K_e}(\boldsymbol{x})|_{\Gamma_f} = \sum_{i=0}^{p} u_i\psi_{i,K_f}(\boldsymbol{x})|_{\Gamma_f} \tag{3.26}$$

where

$$U_i = \langle\varphi_{i,K_e}, u_h|_{K_e}\rangle\ ,\ u_i = \langle\varphi_{i,K_f}, u_h|_{K_f}\rangle \tag{3.27}$$

are values of degrees of freedom obtained for elements K_e and K_f and φ_{i,K_e}, φ_{i,K_f} are the degrees of freedom understood as linear functionals on $X_h(K_e)$ and $X_h(K_f)$. Introducing (3.25) into (3.26), we obtain:

$$\sum_{i=0}^{p}\sum_{j=0}^{p} U_i\ {}^dR_{ij}\psi_{j,K_f}(\boldsymbol{x})|_{\Gamma_f} = \sum_{i=0}^{p} u_i\psi_{i,K_f}(\boldsymbol{x})|_{\Gamma_f}. \tag{3.28}$$

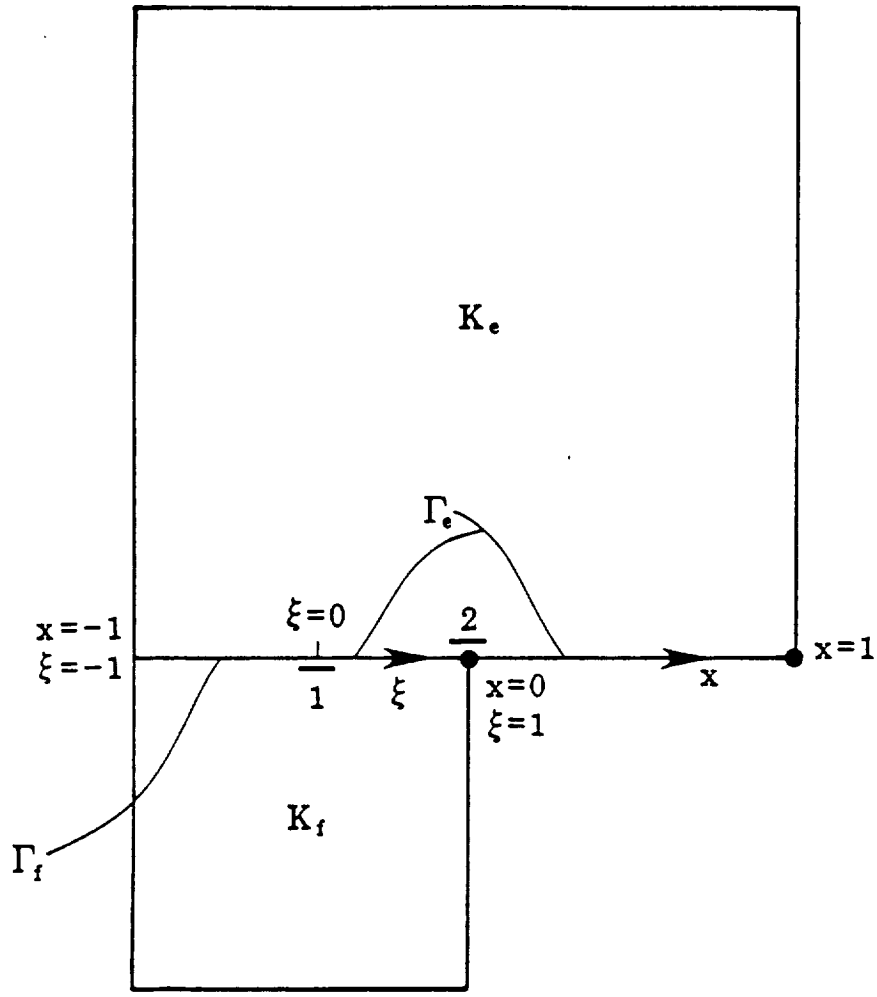


Figure 3.6: Two adjacent elements in a 1-irregular mesh.

Since representation of any function from $\mathcal{P}^p(\Gamma_f)$ in terms of the basis $\psi_{j,K_f}(\boldsymbol{x})\ |_{\Gamma_f}$, $j = 0,\ldots,p$ is unique, we finally find that:

$$u_i = \sum_{j=0}^{p} U_j\ {}^dR_{ji}\ ,\ i = 0,\ldots,p \tag{3.29}$$

or, in view of (3.27),

$$\langle \varphi_{i,K_f}, u_h|_{K_f}\rangle = \sum_{j=0}^{p} \langle \varphi_{j,K_e}\ {}^dR_{ji}, u_h|_{K_e}\rangle \quad i = 0,\ldots,p, \tag{3.30}$$

Formula (3.30) expresses the relation between degrees of freedom of K_e and K_f necessary and sufficient for continuity of approximation along Γ_f. Necessity was shown above. Sufficiency follows from (3.25) and (3.30):

For any u_h such that $u_h|_{K_e} \in X_h(K_e)$, $u_h|_{K_f} \in X_h(K_f)$, not necessarily continuous, we have

$$\begin{aligned}
(u_h|_{K_e})|\Gamma_f &= \sum_{i=0}^{p} \langle \varphi_{i,K_e}, u_h|_{K_e}\rangle \psi_{i,K_e}|\Gamma_f = \\
&= \sum_{i=0}^{p} \langle \varphi_{i,K_e}, u_h|_{K_e}\rangle \sum_{j=0}^{p} {}^dR_{ij}\psi_{j,K_f}|\Gamma_f \\
(u_h|_{K_f})|\Gamma_f &= \sum_{i=0}^{p} \langle \varphi_{i,K_f}, u_h|_{K_f}\rangle \psi_{i,K_f}|\Gamma_f = \\
&= \sum_{i=0}^{p}\sum_{j=0}^{p} \langle \varphi_{j,K_e}\ {}^dR_{ji}, u_h|_{K_e}\rangle \psi_{i,K_e}|\Gamma_f
\end{aligned}$$

i.e., $(u_h|_{K_e})|\Gamma_f = (u_h|_{K_f})|\Gamma_f$ which means that u_h is continuous along Γ_f. Relation (3.30) is referred to as equations of constraints.

The main idea of a constrained approximation is to replace the degrees of freedom of K_f involved in (3.30) by a new set $\tilde{\varphi}_{i,K_f}$, $i = 0,\ldots,p$ related to the old φ_{i,K_f} by the matrix ${}^dR_{ij}$:

$$\varphi_{j,K_f} = \sum_{i=0}^{p} \tilde{\varphi}_{i,K_f}\ {}^dR_{ij} \tag{3.31}$$

Then the continuity condition (3.30) in terms of $\tilde{\varphi}_{j,K_f}$ becomes:

$$\langle \tilde{\varphi}_{j,K_f}, u_h|_{K_f}\rangle = \langle \varphi_{j,K_e}, u_h|_{K_e}\rangle \tag{3.32}$$

i.e., the condition which is formally the same as the condition for continuous approximation on regular meshes. As a consequence, an element K_f equipped with degrees of freedom $\tilde{\varphi}_{j,K_f}$

can be treated in all procedures of finite element code involving continuity (like assembling stiffness matrix) *exactly the same way* as elements of a regular mesh.

The degrees of freedom $\tilde{\varphi}_{j,K_f}$ defined by (3.31) have a simple interpretation. To compute $\langle \tilde{\varphi}_{j,K_f}, \psi \rangle$ for any $\psi \in X_h(K_f)$ it suffices to find any continuous extension $u_h(\boldsymbol{x})$ such that $u_h|_{K_f} = \psi$, $u_h|_{K_e} \in X_h(K_e)$ for which (3.32) holds, i.e.:

$$\langle \tilde{\varphi}_{j,K_f}, \psi \rangle = \langle \varphi_{j,K_e},\ u_h|_{K_e} \rangle \tag{3.33}$$

In fact, since the action of φ_{j,K_e} involves only values of $u_h|_{K_e}$ along the side Γ_e, we need only extend ψ to Γ_e and such an extension, since it must be a polynomial, is unique:

$$\begin{cases} \psi|_{\Gamma_f} = w(s) = \text{polynomial of } s, \\ \\ \text{extension of } \psi \text{ to } \Gamma_e \overset{\text{def.}}{=} w(s) \end{cases} \tag{3.34}$$

and moreover

$$\langle \tilde{\varphi}_{j,K_f}, \psi \rangle = \langle \varphi_{j,K_e},\ u_h|_{K_e} \rangle = \begin{cases} w(s = \pm 1)\ , & j = 0, 1 \\ \\ \lambda_j^{-1} \dfrac{d^j w}{ds^j}(0)\ , & j = 2, \ldots, p \end{cases} \tag{3.35}$$

Illustration of $w(s)$ and degrees of freedom $\tilde{\varphi}_{j,K_f}$ is given in Fig. 3.7a.

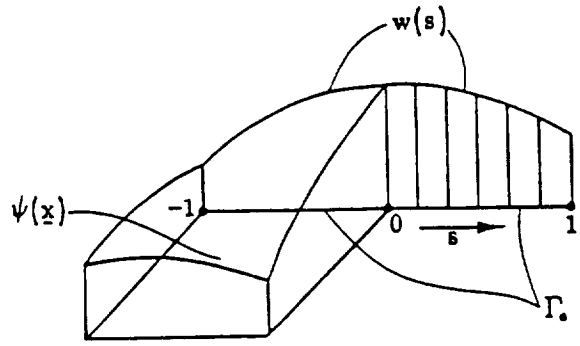
We observe that an additional advantage of this approach is that $\tilde{\varphi}_{j,K_f}$ can be associated with certain points: endpoints and a midpoint of Γ_e, i.e., points coinciding with nodes of the anticipated neighbor. We call these nodes active (actual) nodes of an element while nodes corresponding to original φ_{i,K_f} constrained nodes (Fig. 3.7b).

In case an element is adjacent to two larger neighbors, the transformation of degrees of freedom analogous to (3.31) should be performed for the other side of the element as well.

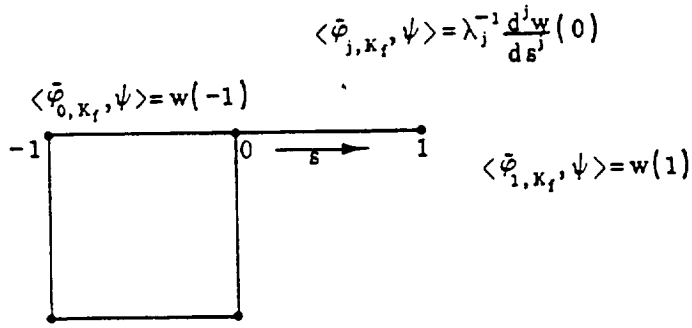
Interpretation of $\tilde{\varphi}_{j,K_f}$ shown in Fig. 3.7a suggests a convenient and intuitive way of treating a constrained element as if its domain included not only the original square but also the adjacent sides of bigger neighbors. This idea gives also a very easy interpretation of assembling constrained elements into a mesh: we connect appropriate actual nodes of elements as for regular meshes (Fig. 3.8).

We conclude this section with a discussion of the problem of evaluating the shape functions dual to the new set of degrees of freedom. Let us write the transformation of degrees of freedom resulting from applying formula (3.31) to one or two sides of a constrained element in the general form:

$$\begin{cases} \varphi_i \ = \ \displaystyle\sum_{j=I(i)} \tilde{\varphi}_j R_{ji} \ , & i \in N^c \\ \\ \varphi_i \ = \ \tilde{\varphi}_i \ , & i \in N^a \end{cases} \tag{3.36}$$



a)



b)

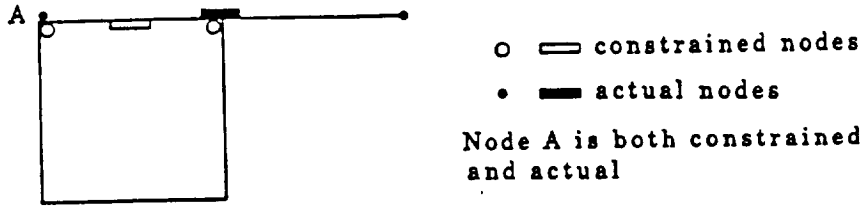


Figure 3.7: A constrained element: (a) illustration of $w(s)$ and degrees of freedom $\tilde{\varphi}_{j,K_f}$; (b) active (actual) and constrained nodes.

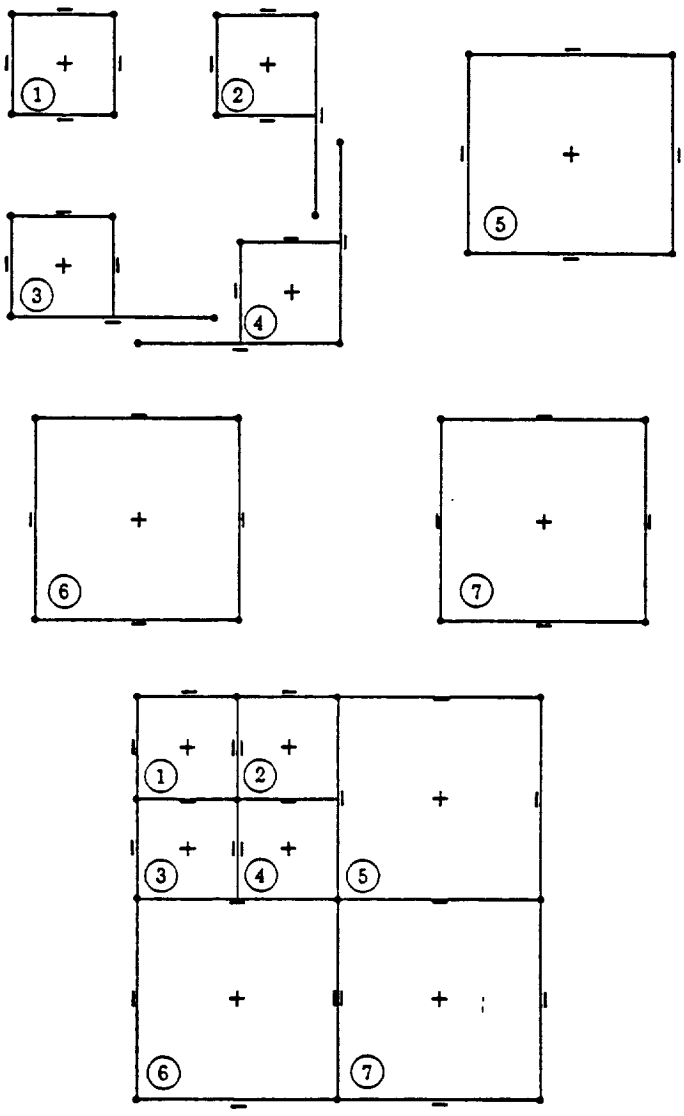


Figure 3.8: Assembling constrained elements into a mesh.

where N^c is a set of indices denoting degrees of freedom associated with constrained sides (i.e., appearing on the left hand side of (3.31) applied to one or two constrained sides of the element, whichever case is considered), N^a-indices of the remaining degrees of freedom, $I(i)$-indices of new degrees of freedom $\tilde{\varphi}_j$ contributing to a given φ_i, $i \in N^c$. $(3.36)_2$ expresses the fact that φ_i's not involved in constraints remain unchanged. Relation (3.36) is invertible since (3.31) is invertible. The change of basis in any linear space is given by:

$$\begin{cases} \varphi_i = \sum_{j=1}^{N} \tilde{\varphi}_j A_{ji} \\ \\ \tilde{\varphi}_i = \sum_{j=1}^{N} \varphi_j A^{-1}_{ji} \end{cases} \tag{3.37}$$

This implies a change of dual basis according to:

$$\begin{cases} \tilde{\psi}_i = \sum_{j=1}^{N} A_{ij} \psi_j \\ \\ \psi_i = \sum_{j=1}^{N} A^{-1}_{ij} \tilde{\psi}_j \end{cases} \tag{3.38}$$

Formulas (3.36) correspond to $(3.37)_1$. This means that expressions for the new base shape functions $\tilde{\psi}_i$ can be found by simply transposing the linear relation (3.36) defining $\tilde{\varphi}_i$.

Transposing (3.36) to obtain $\tilde{\psi}_i$ results in the following formula:

$$\tilde{\psi}_i = \begin{cases} \sum_{j \in S(i)} R_{ij} \psi_j \quad , \quad i \in N^c, \\ \\ \psi_i \qquad\quad , \quad i \in N^a. \end{cases} \tag{3.39}$$

where $S(i) = \{j \in N^c : \tilde{\varphi}_i$ contributes to φ_j in (3.36), i.e., $i \in I(j)\}$.

As a simple example of constraints and application of presented formulas consider a bilinear element with two constrained sides, i.e., adjacent to two larger neighbors, Fig. 3.9. In this case values of degrees of freedom are just values of shape functions at the corner nodes and conditions for continuity have an obvious form: a value of a shape function at a "hanging" node must be equal to the average of the corner values of the shape function of the larger neighbor (Fig. 3.10). Hence, recalling the interpretation of $\tilde{\varphi}_i$'s (Fig. 3.7a), we

write equations (3.36) for element considered as follows:

$$\begin{cases} \varphi_1 = \tilde{\varphi}_1 \\ \varphi_2 = \frac{1}{2}\tilde{\varphi}_1 + \frac{1}{2}\tilde{\varphi}_2 \\ \varphi_4 = \frac{1}{2}\tilde{\varphi}_1 + \frac{1}{2}\tilde{\varphi}_4 \\ \varphi_3 = \tilde{\varphi}_3 \end{cases} \tag{3.40}$$

In this case, $N^c = \{1,2,4\}$, $N^a = \{3\}$. The matrix form of (3.40) is:

$$[\varphi_1, \varphi_2, \varphi_4, \varphi_3] = \cdot [\tilde{\varphi}_1, \tilde{\varphi}_2, \tilde{\varphi}_4, \tilde{\varphi}_3] \begin{vmatrix} 1 & \frac{1}{2} & \frac{1}{2} & \\ & \frac{1}{2} & & \\ & & \frac{1}{2} & \\ & & & 1 \end{vmatrix} \tag{3.41}$$

According to $(3.38)_1$ the base shape functions $\tilde{\psi}_i$ are given by

$$\begin{vmatrix} \tilde{\psi}_1 \\ \tilde{\psi}_2 \\ \tilde{\psi}_4 \\ \tilde{\psi}_3 \end{vmatrix} = \begin{vmatrix} 1 & \frac{1}{2} & \frac{1}{2} & \\ & \frac{1}{2} & & \\ & & \frac{1}{2} & \\ & & & 1 \end{vmatrix} \cdot \begin{vmatrix} \psi_1 \\ \psi_2 \\ \psi_4 \\ \psi_3 \end{vmatrix} \tag{3.42}$$

or

$$\begin{cases} \tilde{\psi}_1 = \psi_1 + \frac{1}{2}\psi_2 + \frac{1}{2}\psi_4 \\ \tilde{\psi}_2 = \frac{1}{2}\psi_2 \\ \tilde{\psi}_4 = \frac{1}{2}\psi_4 \\ \tilde{\psi}_3 = \psi_3 \end{cases} \tag{3.43}$$

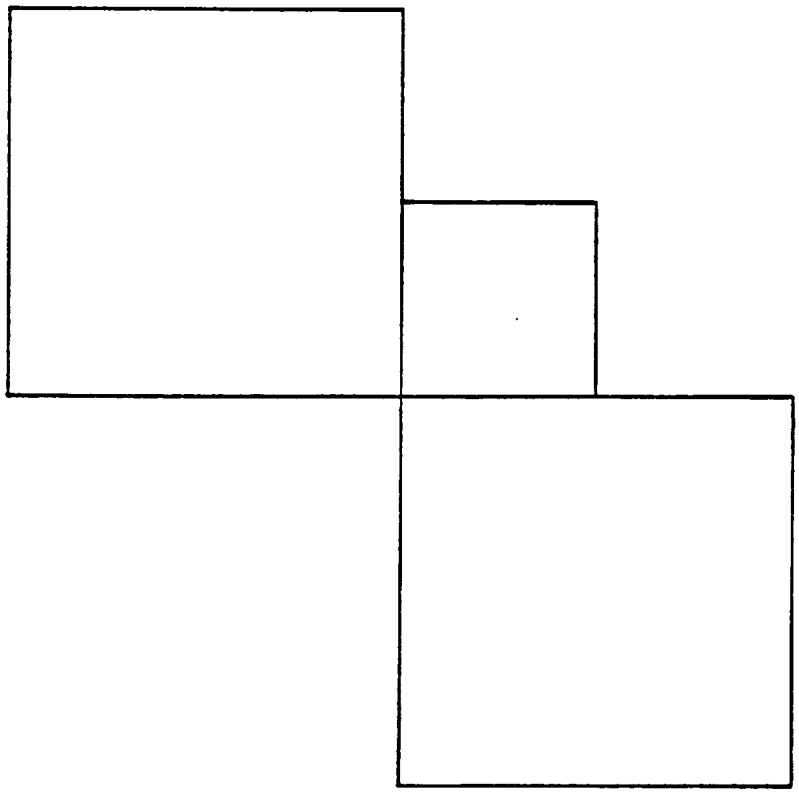


Figure 3.9: An element adjacent to two larger neighbors.

Figure 3.11 presents the constrained base shape functions $\tilde{\psi}_i$. Relation (3.43) could also be formally obtained using the general expression (3.39).

3.9 Calculation of the Element Load Vector and Stiffness Matrix

For simplicity, we restrict ourselves to the case of a single equation. In the case of systems the same procedure is applied for every linear form (3.7) and bilinear form (3.4).

The element load vector and stiffness matrix are defined as:

$$\begin{cases} \tilde{b}_i &= L_{h,K}\left(\tilde{\psi}_i\right), \\ \\ \tilde{B}_{ij} &= B_{h,K}\left(\tilde{\psi}_i, \tilde{\psi}_j\right), \end{cases} \tag{3.44}$$

$i, j = 1, \ldots, N$, N being the number of degrees of freedom of an element, $\tilde{\psi}_i$ base shape functions corresponding to actual degrees of freedom $\tilde{\varphi}_i$.

Since constrained shape functions $\tilde{\psi}_i$ are related to the usual shape functions ψ_i by a linear transformation (3.39), $\tilde{b}_i$ and $\tilde{B}_{ij}$ can be expressed as follows:

$$\begin{cases} \tilde{b}_i &= \displaystyle\sum_{j \in S(i)} R_{ij} L_{h,K}(\psi_j) \quad \text{for} \quad i \in N^c, \\ \\ \tilde{b}_i &= L_{h,K}(\psi_i) \qquad\qquad \text{for} \quad i \in N^a, \end{cases}$$

$$\begin{cases} \tilde{B}_{ij} &= \displaystyle\sum_{k \in S(i)} \sum_{\ell \in S(j)} R_{ik} R_{j\ell} B_{h,K}(\psi_k, \psi_l) \quad \text{if} \quad i, j \in N^c, \\ \\ \tilde{B}_{ij} &= \displaystyle\sum_{k \in S(i)} R_{ik} B_{h,K}(\psi_k, \psi_j) \qquad \text{if} \quad i \in N^c, j \in N^a, \\ \\ \tilde{B}_{ij} &= \displaystyle\sum_{k \in S(j)} R_{jk} B_{h,K}(\psi_i, \psi_k) \qquad \text{if} \quad i \in N^a, j \in N^c, \\ \\ \tilde{B}_{ij} &= B_{h,K}(\psi_i, \psi_j) \qquad \text{if} \quad i, j \in N^a \end{cases} \tag{3.45}$$

where $L_{h,K}$, $B_{h,K}$ are restrictions of forms (3.7) and (3.4) to the element K, sets of indices N^c, N^a, $S(i)$ were defined in (3.36) and (3.39).

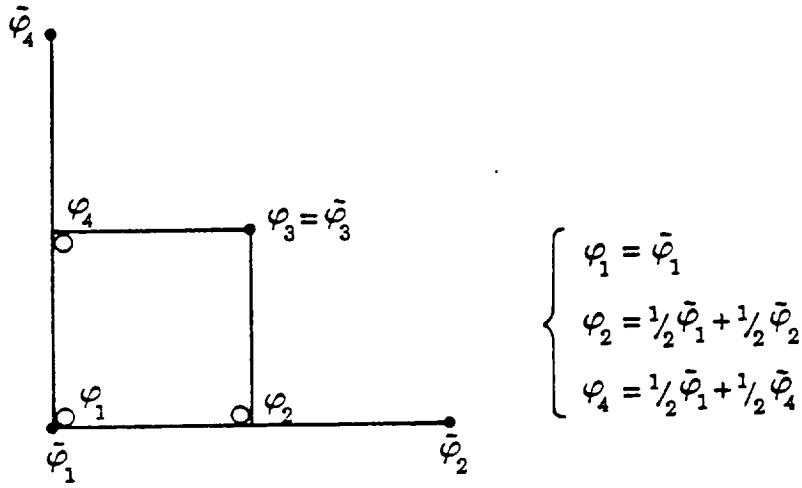


Figure 3.10: Conditions for continuity for a constrained bilinear element.

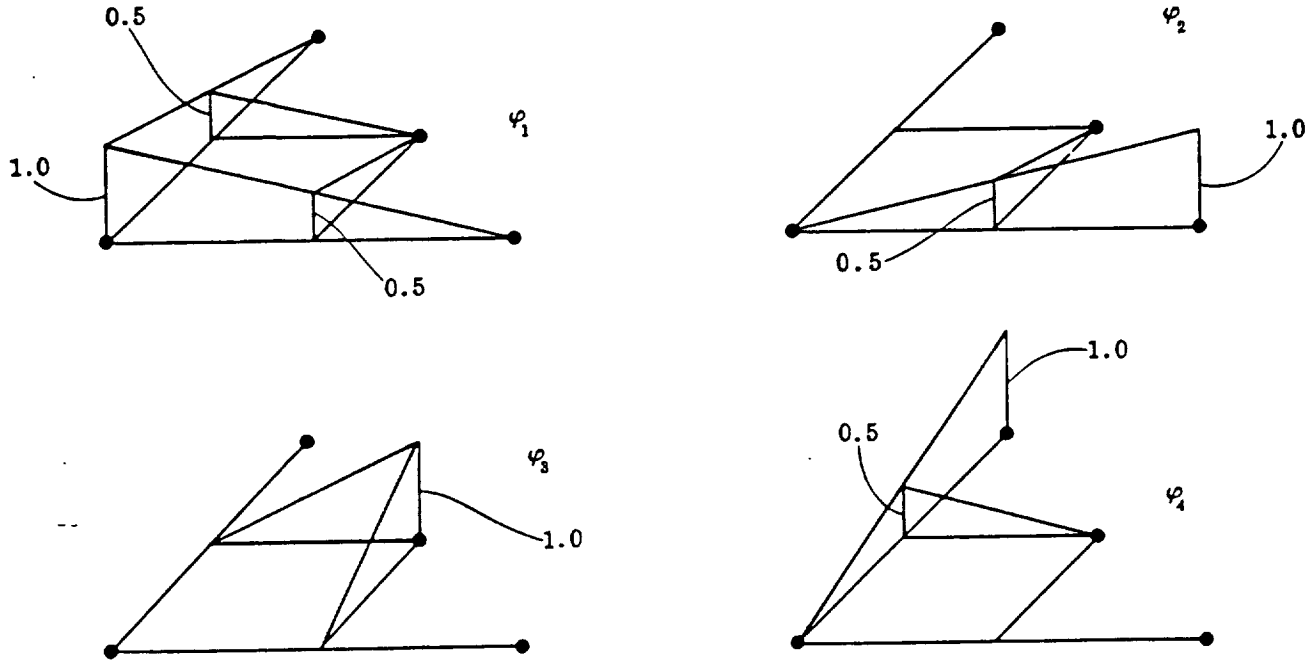


Figure 3.11: Constrained base shape functions $\tilde{\psi}_i$.

3.10 Constraints in the One–Dimensional Case

In the case of irregular meshes, continuity has to be enforced by means of the constrained approximation. To fix ideas, consider the generic, one–dimensional case shown in Fig. 3.12. The approximation on the small elements $[-1,0]$ and $[0,1]$ must match the approximation on the large element $[-1,1]$.

We first choose the scaling factors λ_p in (3.18) in such a way that the corresponding shape functions for the one–dimensional master element have the following form

$$\left.\begin{array}{rcl} \chi_0 &=& \frac{1}{2}(1-\xi) \\ \chi_1 &=& \frac{1}{2}(1+\xi) \\ \chi_p(\xi) &=& \begin{cases} \xi^p - 1 & p = 2,4,6,\ldots \\ \xi^p - \xi & p = 3,5,7,\ldots \end{cases} \end{array}\right\} \tag{3.46}$$

Assume next that all degrees of freedom for the large element are active. The question is: what degrees of freedom must the small elements take on in order that the functions supported on the two small elements exactly coincide with shape functions of the large element?

From the fact that (3.18) is a dual basis to (3.46), we get

$$\varphi_p(\chi_p) = \frac{1}{\lambda_p} p! = 1 \quad p = 2,3,\ldots \tag{3.47}$$

and therefore $\lambda_p = p!$.

The transformation map from $[-1,1]$ onto $[-1,0]$ is of the form

$$x = -\frac{1}{2} + \frac{1}{2}\xi \tag{3.48}$$

with inverse $\xi = 2x + 1$. This yields the following formulas for the shape functions ${}^{\ell}\chi_p, p = 0,1,2,\ldots$ For the (left-hand side) element $[-1,0]$ (recall definition of a subparametric element).

$$\begin{array}{rcl} {}^{\ell}\chi_0(x) &=& -x \\ {}^{\ell}\chi_1(x) &=& x+1 \\ {}^{\ell}\chi_p(x) &=& 1-(2x+1)^p \quad p = 2,4,6,\ldots \\ {}^{\ell}\chi_p(x) &=& (2x+1)^p - (2x+1) \quad p = 3,5,7,\ldots \end{array} \tag{3.49}$$

and the corresponding formulas for the degrees of freedom are

$$<^l \varphi_0, u> \;=\; u(-1)$$

$$<^l \varphi_1, u> \;=\; u(0)$$

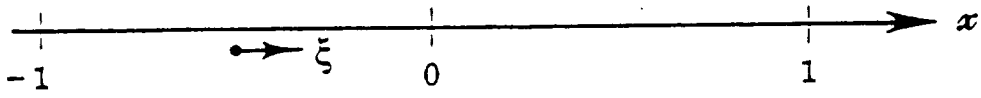


Figure 3.12: Derivation of the constraint coefficients.

$$<^l \varphi_p, u >= \frac{1}{2^p p!} \frac{d^p u}{dx^p} \left(-\frac{1}{2}\right) \quad p = 2, 3, \ldots \tag{3.50}$$

Now let $u(x)$ ($x = \xi$ for the master element) be any function defined by the shape functions on $[-1, 1]$, i.e.,

$$u(x) = \sum_{q=0}^{k} \varphi_q(u) \chi_q(x) \tag{3.51}$$

In order to represent $u(x)$ for $x \in [-1, 0]$ in terms of the shape functions on $[-1, 0]$, we have to calculate the values of the degrees of freedom (3.50). We get

$$\begin{aligned}
<^\ell \varphi_0, u > &= \varphi_0(u) <^\ell \varphi_0, \chi_0 > + \sum_{q=1}^{k} \varphi_q(u) <^l \varphi_0, \chi_q > \\
&= < \varphi_0, u > \\
<^l \varphi_1, u > &= \varphi_0(u) <^l \varphi_1, \chi_0 > + \varphi_1(u) <^l \varphi_1, \chi_1 > \\
&+ \sum_{q=2}^{k} \varphi_q(u) <^l \varphi_1, \chi_q > \\
&= \frac{1}{2} < \varphi_1, u > + \sum_{q=2}^{k} {}^l R_{q1} < \varphi_q, u >
\end{aligned} \tag{3.52}$$

where

$$\begin{aligned}
{}^l R_{q1} &= <^l \varphi_1, \chi_q > \\
&= \begin{cases} 1 & \text{if } q \text{ is even} \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{3.53}$$

For $p \geq 2$,

$$\begin{aligned}
<^l \varphi_p, u > &= \varphi_0(u) <^l \varphi_p, \chi_0 > + \sum_{q=1}^{k} \varphi_q(u) <^l \varphi_p, \chi_q > \\
&= 0 + \sum_{q=1}^{k} {}^l R_{qp} < \varphi_q, u >
\end{aligned}$$

where

$$\begin{aligned}
{}^l R_{qp} &= <^l \varphi_p, \chi_q > \\
&= \begin{cases} 0 & \text{for } q < p \\ \frac{1}{2^q}(-1)^{p+q} \binom{q}{p} = \frac{(-1)^{p+q}}{2^q} \frac{q!}{p!(q-p)!} & \text{for } q \geq p \end{cases}
\end{aligned} \tag{3.54}$$

The same procedure applied to the right-hand side element [0,1] yields the following:

The transformation from $[-1,1]$ *onto* $[0,1]$

$$x = \frac{1}{2} + \frac{1}{2}\xi \tag{3.55}$$

with inverse $\xi = 2x - 1$.

The shape functions ${}^r\chi_p, p = 0,1,2,\ldots$:

$$ {}^r\chi_0(x) = 1 - x \tag{3.56}$$

$$ {}^r\chi_1(x) = x \tag{3.57}$$

$$ {}^r\chi_p(x) = \begin{cases} 1-(2x-1)^p & p \text{ even} \\ (2x-1)^p - (2x-1) & p \text{ odd} \end{cases} \tag{3.58}$$

The degrees of freedom ${}^r\varphi_p$:

$$< {}^r\varphi_0, u > = u(0)$$

$$< {}^r\varphi_1, u > = u(1)$$

$$< {}^r\varphi_p, u >= \frac{1}{2^p p!}\frac{d^p u}{dx^p}\left(\frac{1}{2}\right) \tag{3.59}$$

The constraints

$$< {}^r\varphi_0, u >= \frac{1}{2} < \varphi_0, u > + \frac{1}{2} < \varphi_1, u > + \sum_{q=2}^{k} {}^rR_{q0} < \varphi_q, u > \tag{3.60}$$

where

$$ {}^rR_{q0} = \begin{cases} 1 & \text{if } q \text{ is even} \\ 0 & \text{otherwise} \end{cases} \tag{3.61}$$

$$< {}^r\varphi_1, u >=< \varphi_1, u > \tag{3.62}$$

and for $p \geq 2$

$$< {}^r\varphi_p, u >=< \sum_{q=2}^{k} {}^rR_{qp} < \varphi_q, u > \tag{3.63}$$

where

$$ {}^{r}R_{qp} = \langle {}^{r}\varphi_p, \chi_q \rangle \tag{3.64} $$

$$ = \begin{cases} 0 & \text{for } q < p \\ \dfrac{1}{2^q}\binom{q}{p} & \text{for } q \geq p \end{cases} \tag{3.65} $$

Finally, the conditions for matching the approximation on one–dimensional master element $[-1,1]$ and on the two small elements $[-1,0]$, $[0,1]$ can be rewritten as:

$$ {}^{d}\varphi_i = \sum_{j=0}^{p} \varphi_j \, {}^{d}R_{ji} \, , \quad i = 0, \ldots, p \tag{3.66} $$

where the superscript $d = l$ (left) or r (right) distinguishes between the two small elements. Formula (3.66) relating degrees of freedom of the two elements is equivalent to a corresponding relation between the base shape functions ${}^{d}\chi_i$ and χ_i dual to ${}^{d}\varphi_i$ and φ_i:

$$ \chi_i = \sum_{j=0}^{p} {}^{d}R_{ij} \, {}^{d}\chi_j \, , \quad i = 0, \ldots, p \tag{3.67} $$

Arrays ${}^{l}R_{qp}$ and ${}^{r}R_{qp}$, $q, p = 0, \ldots, 5$ are presented in Fig. 3.13.

3.11 Constraints for Two–Dimensional Subparametric Elements

Since the shape functions for the 2-D master element are defined as tensor products of the 1-D functions, the results for the 1-D case hold exactly in the same form in the 2-D situation, the only difference being that the calculated constraint equations have to be applied to the proper degrees of freedom (see 3.31). It follows from the definition of the subparametric elements that the constraints coefficients are *exactly the same*, even when the elements have curved boundaries. This follows from the fact that the shape functions' behavior in a subparametric element on a part of its boundary depends exclusively upon the deformation of the part of the boundary, and therefore, any relation defined for the shape functions in the generic situation carries over immediately to the case of two small elements sharing an edge with a large element, so long as the deformation of the edge is identical in all three elements. The situation is illustrated in Fig. 3.14.

3.12 Constrained Approximation in a Three–Dimensional Case

A three–dimensional constrained approximation exploits basically the same concepts as we developed for the two–dimensional case. The constraints, however, in this case have a more

$$
{}^{l}R_{qp} = \begin{bmatrix}
1 & 1/2 & & & & & \\
 & 1/2 & & & & & \\
 & -1 & 1/4 & & & & \\
 & 0 & -3/8 & 1/8 & & & \\
 & -1 & 6/16 & -4/16 & 1/16 & & \\
 & 0 & -10/32 & 10/32 & -5/32 & 1/32 & \\
 & -1 & 15/64 & -20/64 & 15/64 & -6/64 & 1/64
\end{bmatrix}
$$

$$
{}^{r}R_{qp} = \begin{bmatrix}
1/2 & & & & & & \\
1/2 & 1 & & & & & \\
-1 & & 1/4 & & & & \\
0 & & 3/8 & 1/8 & & & \\
-1 & & 6/16 & 4/16 & 1/16 & & \\
0 & & 10/32 & 10/32 & 5/32 & 1/32 & \\
-1 & & 15/64 & 20/64 & 15/64 & 6/64 & 1/64
\end{bmatrix}
$$

Figure 3.13: The constraint coefficients for a sixth order approximation. The unfilled coefficients are zero.

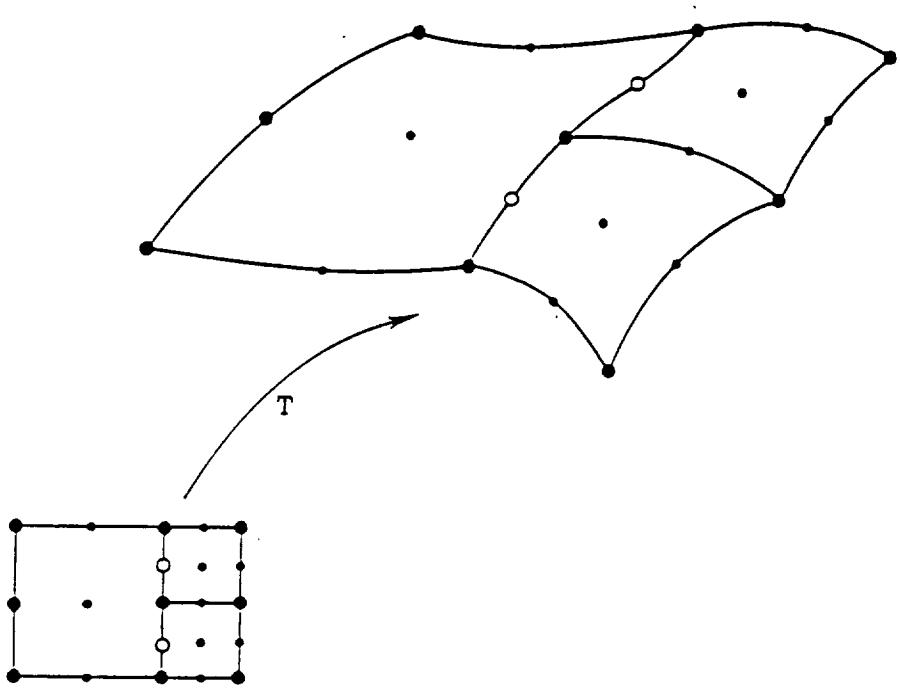


Figure 3.14: Illustration of the constraints for the subparametric elements. (after [9])

complex form. Moreover, there are two different types of constraints: one resulting from constraining only an edge of the element and the second resulting from constraining the whole wall of the element.

(a) an element may have a larger neighbor across the wall, Fig. 3.15a

(b) the neighbors across walls are of the same size as the element considered, but the neighbor across an edge is larger, Fig. 3.15b

These two situations must be considered separately.

3.13 Constraints for a Wall

We construct equations of constraints and transformation to new degrees of freedom and base shape functions in essentially the same way as we did in the two–dimensional case.

We consider two adjacent cube elements K_e and K_f of a 1-irregular mesh, one of them, K_f, two times smaller than K_e. The wall W_f of K_f is attached to one quarter of the wall W_e of the larger neighbor, Fig. 3.16. The first condition for continuity of approximation is that spaces of shape functions $X_h(K_e)$ and $X_h(K_f)$ if restricted to W_f must be identical:

$$X_h(K_f)|_{W_f} = X_h(K_e)|_{W_f} \tag{3.68}$$

and it is satisfied since we accepted the "maximum rule".

In the reasoning below we use two–indices notation for shape functions of K_f and K_e which do not vanish identically on W_f : ψ_{ij,K_e} and ψ_{ij,K_f}. Restrictions of these functions to W_f are of the following form (Fig. 3.15)

$$\begin{cases} \psi_{ij,K_e|W_f} = \chi_i(x) \cdot \chi_j(y) & i,j = 0,\ldots,p \\ \psi_{ij,K_f|W_f} = {}^{d_1}\chi_i(\xi) \cdot {}^{d_2}\chi_i(\eta) & i,j = 0,\ldots,p \end{cases} \tag{3.69}$$

where χ_i and ${}^{d_1}\chi_i$, ${}^{d_2}\chi_i$ are one–dimensional hierarchical shape functions considered in the section on "Constraints in One–Dimensional Case," d_1, d_2 indicate which of the two generic situations considered there should be applied.

Functions $\psi_{ij,K_e|W_f}$, $\psi_{ij,K_f|W_f}$ constitute two bases of spaces $X_{h(K_f)|W_f} = X_h(K_e)|_{W_f}$, therefore there must exist a unique linear invertible relation between them. Using the transformation (3.67) between $\chi_i(x)$ and ${}^{d_1}\chi(\xi)$, and the same for the y-direction we easily find that this relation is of the form:

$$\psi_{i,jK_e|W_f} = \sum_{k=0}^{p}\sum_{\ell=0}^{p} {}^{d_1}R_{ik}\ {}^{d_2}R_{j\ell}\psi_{k\ell,K_f|W_f} \tag{3.70}$$

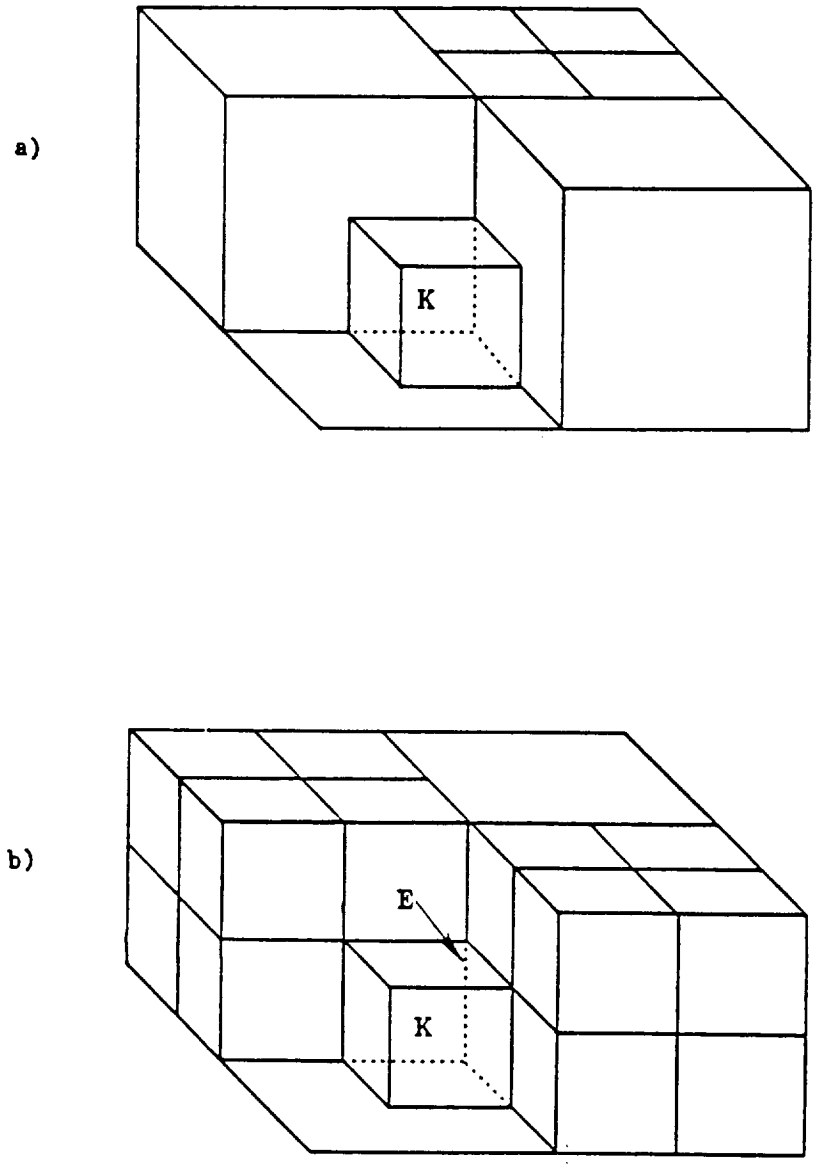


Figure 3.15: Three-dimensional constrained elements: (a) an element with constrained walls; (b) an element with a constrained edge.

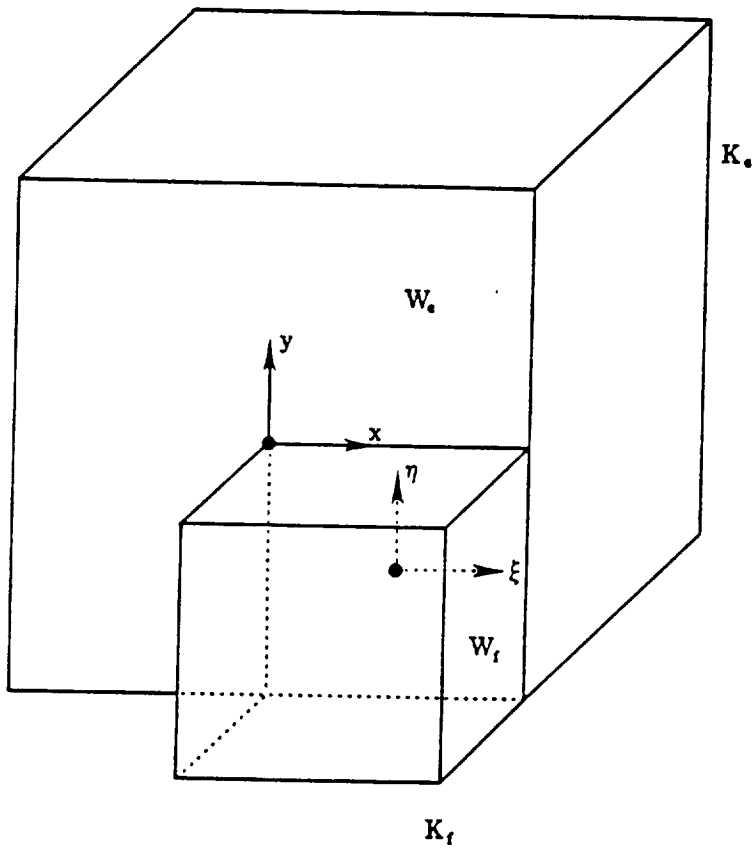


Figure 3.16: Two adjacent three–dimensional elements ina 1-irregular mesh having a common wall.

which is analogous to formula (3.25) obtained for two–dimensional case.

By exactly the same reasoning as in that case we can show that introducing new degrees of freedom $\tilde{\varphi}_{ij,K_f}$ by

$$\varphi_{ij,K_f} = \sum_{k=0}^{p}\sum_{\ell=0}^{p} \tilde{\varphi}^{d_1}_{k\ell,K_f} R^{d_2}_{ki} R_{\ell j} \tag{3.71}$$

we obtain the necessary and sufficient condition for continuity of approximation along the wall of the form:

$$\langle \tilde{\varphi}_{ij,K_f}, u_{h|K_f} \rangle = \langle \varphi_{ij,K_e}, u_{h|K_e} \rangle \tag{3.72}$$

for every u_h such that $u_{h|K_e} \in X_h(K_e)$, $u_{h|K_f} \in X_h(K_f)$.

3.14 Constraints for an Edge

Consider now a situation when a smaller element K_f has a larger neighbor across the edge E_f even though the neighbors across the walls adjacent to E_f are of the same size as K_f, Fig. 3.17. In this case only the shape functions associated with the edge E_f are involved in constraints.

Let ψ_{i,K_e}, ψ_{i,K_f} be the base shape functions of K_e and K_f which do not identically vanish on E_f. Again, continuity of approximation requires that:

$$X_h(K_e)|_{E_f} = X_h(K_f)|_{E_f}. \tag{3.73}$$

The two sets of functions $\psi_{i,K_e|E_f}$, $\psi_{i,K_f|E_f}$ are the basis of the above spaces, and since:

$$\begin{cases} \psi_{i,K_e|E_f} &= \chi_i(x) \\ \psi_{i,K_f|E_f} &= {}^d\chi_i(\xi) \end{cases} \tag{3.74}$$

and $\chi_i(x)$, ${}^d\chi_i(\xi)$ are related by (3.67), we find that

$$\psi_{i,K_e|E_f} = \sum_{j=0}^{p} {}^dR_{ij}\psi_{j,K_f|E_f}. \tag{3.75}$$

By the arguments used before we introduce a new set of degrees of freedom $\tilde{\varphi}_{i,K_f}$ such that:

$$\varphi_{i,K_f} = \sum_{j=0}^{p} \tilde{\varphi}_{j,K_f}\ {}^dR_{ji} \tag{3.76}$$

and as a consequence we obtain the necessary and sufficient condition for continuity along E_f:

$$\langle \tilde{\varphi}_{i,K_f}, u_{h|K_f} \rangle = \langle \varphi_{i,K_e}, u_{h|K_e} \rangle \tag{3.77}$$

for all u_h such that $u_{h|K_e} \in X_h(K_e)$, $u_{h|K_f} \in X_h(K_f)$.

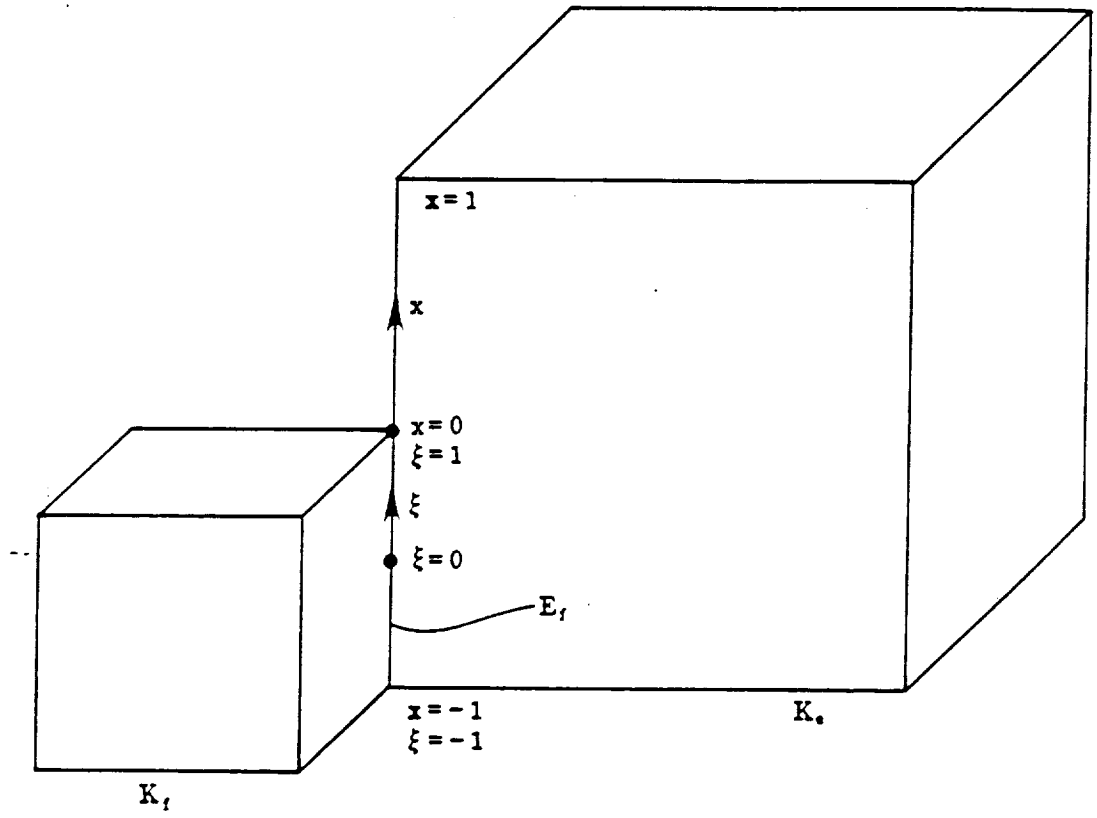


Figure 3.17: Two three–dimensional elements in a 1-irregular mesh with a common edge.

3.15 The Constrained Base Shape Functions

We collect transformations (3.71) and (3.70) obtained for all constrained walls of the element and all constrained edges not belonging to constrained walls. As a result we end up with the relation which, after introducing a uniform one–index numbering of degrees of freedom, can be written in a general form:

$$\begin{cases} \varphi_i = \sum\limits_{j=I(i)} \tilde{\varphi}_j R_{ji} \quad , \quad i \in N^c \\ \\ \varphi_i = \tilde{\varphi}_i \qquad\qquad , \quad i \in N^a \end{cases} \tag{3.78}$$

where N^c is a set of indices denoting degrees of freedom associated with constrained walls and edges of the element, N^a-indices of the remaining degrees of freedom, $I(i)$-indices of $\tilde{\varphi}_i$'s contributing to a given φ_j in (3.78). Coefficients R_{ij} are expressed by ${}^{d_1}R_{ij}$, ${}^{d_2}R_{ij}$ in a way indicated by (3.76) or are just equal to ${}^{d}R_{ij}$ if constraints result from (3.78).

By the arguments used before the base shape functions $\tilde{\psi}_i$ corresponding to $\tilde{\varphi}_i$ are given by a linear transformation transpose to (3.78):

$$\begin{cases} \tilde{\psi}_i = \sum\limits_{j \in S(i)} R_{ij} \psi_j \quad , \quad i \in N^c \\ \\ \tilde{\psi}_i = \psi_i \qquad\qquad , \quad i \in N^a \end{cases} \tag{3.79}$$

where $S(i) = \{j \in N^c \mid i \in I(j)\}$.

3.16 Interpretation of $\tilde{\varphi}_i$. Calculation of the Load Vector and Stiffness Matrix

As in the two–dimensional case, degrees of freedom $\tilde{\varphi}_i$ have a simple interpretation: to compute $\langle \tilde{\varphi}_i, \psi \rangle$, $\psi \in X_h(K)$ we can uniquely extend the function ψ as a polynomial to constrained walls and edges extended such that they coincide with walls and edges of the anticipated larger neighbors. Then, appropriate degree of freedom φ_i of the larger neighbor should be applied for the extension of ψ. All generic situations are presented in Fig. 3.18.

The algorithms for transforming the usual element load vector and stiffness matrix to those corresponding to $\tilde{\varphi}_i$'s are exactly the same as in the two–dimensional case. The only difference is that they use more complex formulas (3.79).

Also, the arguments that we used to extend the concept of constrained approximation to two–dimensional subparametric elements apply in the three–dimensional case.

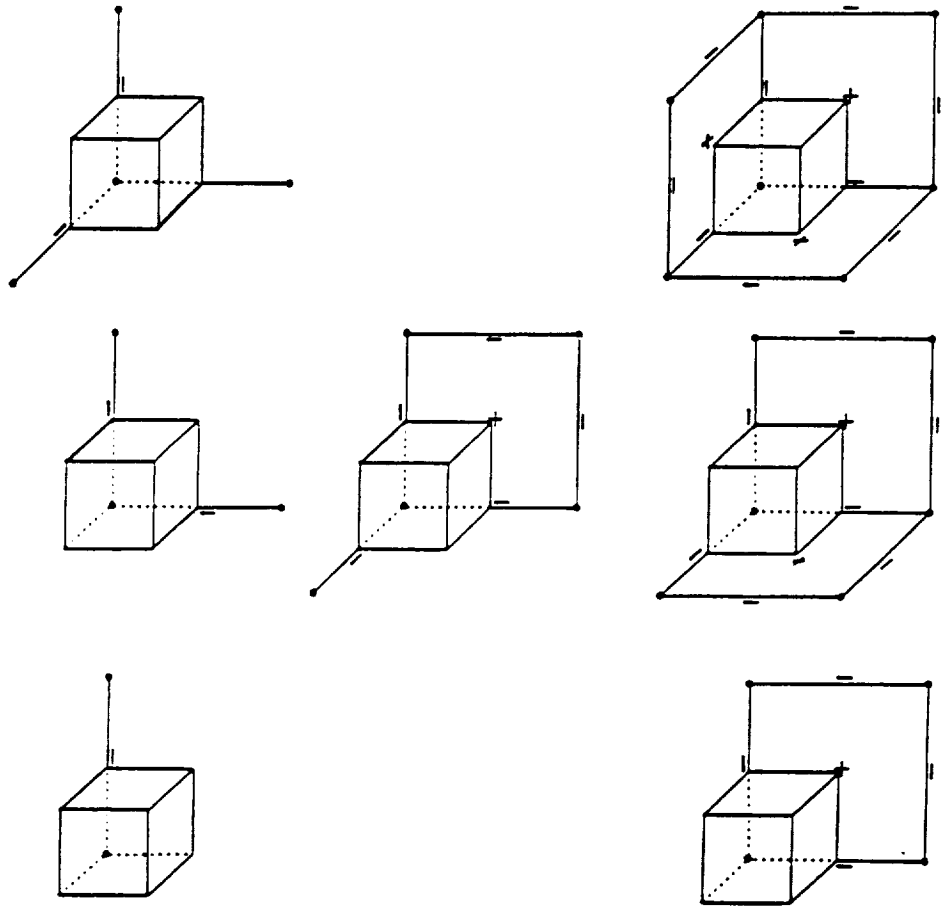


Figure 3.18: Examples of three–dimensional constrained elements.

3.17 Concluding Remarks on Constrained Approximation

- A formal definition of an element states that it is a triple:

$$\{K, X_h(K), \{\varphi_i\}_{i=1,\ldots,N}\}, \tag{3.80}$$

  where $K \subset \mathbb{R}^2(\mathbb{R}^3)$ is a domain of the element, $X_h(K)$ the space of shape functions, $\{\varphi_i\}_{i=1,\ldots,N}$ a set of linear functionals on $X_h(K)$ called degrees of freedom.

  From this point of view the approach that we propose for enforcing continuity on 1-irregular meshes is equivalent to constructing a new element: we define new degrees of freedom.

- A finite element code using constrained elements must include only three non-traditional algorithms:
  - a procedure identifying kinds of constraints for a given element,
  - an algorithm transforming the usual load vector and stiffness matrix to those corresponding to actual degrees of freedom $\tilde{\varphi}_i$,
  - the procedure transforming the finite element solution in terms of $\tilde{\varphi}_i$'s (obtained from the solver) to values of usual degrees of freedom, i.e., the procedure performing the calculations indicated by (3.36) or (3.78).

These three algorithms involve complex logical operations; however, once they are coded, they may be used as "black boxes" by a user not familiar with their content. The rest of the code is unaffected by the constrained approximation and therefore it may be developed in a standard way.

3.18 Some Details Concerning the Data Structure

In the classical finite element method, elements as well as nodes are usually numbered consecutively in an attempt to produce a minimal band within the global stiffness matrix. When the program identifies an element to process its contribution to the global matrices, the minimal information needed is the node numbers associated with the element. Adaptive refinement and unrefinement algorithms require much more information on the mesh structure than the classical assembly process.

First of all, we introduce the notion of a *family*. Whenever an element is refined a new family is created. The original element is called the *father* of the family and the four new elements are called its *sons*. Graphically, the geneology on families can be presented in a family tree structure as illustrated in Fig. 3.19.

An examination of refinement and unrefinement algorithms (see [7] for details) reveals that for a given element NEL, one must have access to the following information:

- element node numbers
- element neighbors
- the three structure information, including:
  - number of the element family
  - number of the father
  - numbers of the sons
  - refinement level (number of generation)

For a given NODE we also require,

- node coordinates
- values of the degrees of freedom associated with the node

In general, some information is stored explicitly in a data base consisting of a number of arrays, some other information is recovered from the data base by means of simple algorithms. A careful balance should be maintained between the amount of information stored (storage requirements) and recovered (time).

The following is a short list of arrays used in the data base:

1. The tree structure is stored in a condensed, family-like fashion [26], [7] in two arrays

   NSON(NRELEI)
   NTREE(5,MAXNRFAM)

   where NRELEI is the number of elements in the initial mesh and MAXNRFAM is the anticipated maximum number of families. For an element NEL of the initial mesh, NSON(NEL) contains its first son number (if there is any). For a family NFAM, NTREE(1,NFAM) contains the number of the father of the family while the other four entries NTREE(2:5, NFAM) are reserved for the "*first-born*" sons of the sons of the family (the first-born "grandsons" of the father).

2. The initial mesh neighbor information is stored explicitly in array

   NEIG(4,NRELEI)

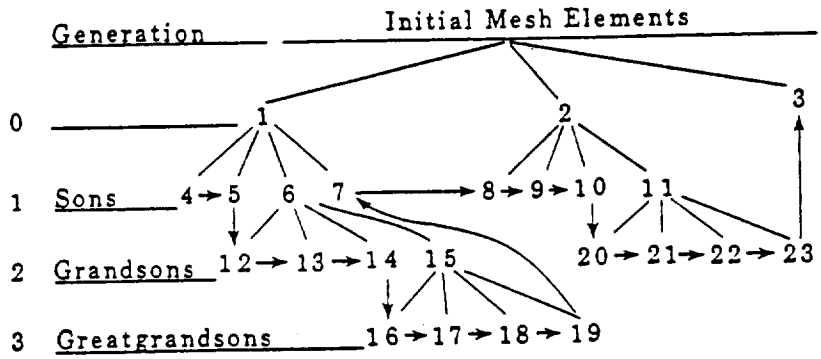


Figure 3.19: A tree structure and the natural order of elements: 4, 5, 12, 13, 14, 16, 17, 18, 19, 7, 8, 9, 10, 20, 21, 22, 23, 3. (after [9])

containing up to four neighbors for each element of the initial mesh (elements adjacent to the boundary may have less neighbors).

3. For every *active* element, up to nine *nicknames* are stored in array

   NODES(9,MAXNRELEM)

   where MAXNRELEM is the anticipated maximum number of elements.

   For a regular node, the nickname is defined as

   NODE*100 + NORDER

   where NODE is the node number and NORDER the order of approximation associated with the node.

   For an irregular node, the nickname is defined as

   NORDER

   where NORDER is again the order of approximation corresponding to the node.

4. For a particular component IEL of a vector-valued solution, the corresponding degrees of freedom are stored sequentially in array

   U(MAXNRDOF,IEL)

   where MAXNRDOF is the anticipated maximal number of degrees of freedom. Two extra integer arrays are introduced to handle the information stored in array U. Array

   NADRES(MAXNRNODE)

   contains for every node, NODE, the address of the *first* from the degrees of freedom corresponding to NODE in array U. If K = NADRES(NODE) is such an address, the address for the next degree of freedom can be found in

   NU(K)

   and so on, until NU(K)=0, which means that the last degree of freedom for a node has been found. The parameter MAXNRNODE above is the anticipated maximal number of nodes.

5. The node coordinates are stored in array XNODE

   XNODE(2,MAXNRNODE)

The rest of the necessary information is *reconstructed* from the data structure by means of simple algorithms. These include:

- calculation of up to eight neighbors for an element
- calculation of *local* coordinates of nine nodes for an element determining its geometry (the irregular nodes coordinates have to be reconstructed by interpolating regular nodes coordinates)
- recovery of the tree–structure related information, e.g., level of refinement, the sons' numbers, etc.
- an algorithm establishing the *natural order of elements*

During the h and p refinements and unrefinements, both elements and nodes are created and deleted in a rather random way. This makes it impossible to denumerate them in a consecutive way, according to their numbers (for instance, as a result of unrefinements some numbers may be simply missing). Thus a new ordering of elements has to be introduced which is based on some scheme other than an element numbers criterion. In the algorithms discussed here, we use "the *natural order of elements*" based on the initial mesh elements ordering and the tree structure. The concept is illustrated in Fig. 3.19. One has to basically follow the tree of elements obeying the order of elements in the initial mesh and the order of sons in a family.

The natural order of elements may serve as a basis for defining an order for nodes and, consequently, for degrees of freedom, when necessary.

For a detailed discussion of the data structure as well as a critical review of different data structures in context of different h-refinement techniques, we refer again to [7].

4 Adaptivity

The main advantage of an *h-p* finite element method is the possibility of adapting the mesh to features of the approximated solution. Adaptivity should lead to enriching the current mesh only where it is needed, i.e., where the accuracy is not sufficient and where the new degrees of freedom cause the best improvement of the solution.

There are two basic steps in the process of adapting the mesh. The first is the *a posteriori* estimation of errors of the current approximation. This estimation provides the necessary local information about the quality of the solution required to drive the adaptive strategy. In the second step, based on the knowledge of the errors, we refine the mesh: break or enrich the elements. The rules for making the decision as to which elements should be refined or enriched play the key role in generating optimal meshes. They are usually referred to as adaptive strategies.

In the following, we first present error estimation techniques which have been implemented for the compressible Navier-Stokes equation. Then we discuss extension of these techniques to calculate the directional adaptation indicator. The *h-p* mesh adaptation strategies are described in the last subsection based on these indications. Note, that the error estimation and *h-p* adaptive techniques were developed in the previous year, while the directional adaptation indicator is a recent development in this project.

4.1 Error Estimation Techniques

In general, there are two major classes of error estimation techniques: interpolation error estimates and residual error estimates. The former group of methods, interpolation methods, provide a rather inexpensive approach for estimating the numerical error. This approach, however, is usually not very accurate and only provides a relative indication of where large errors exist. The latter class of methods, residual methods, are typically much more accurate but are also much more expensive to use. During the course of this project we have experimented with both classes of methods.

4.1.1 Interpolation Error Estimate

This method of error estimation employs well known *a priori* estimates of the interpolation error of finite element approximations. Such estimates are given by the following formula:

$$\|u - u_I\|_{0,K} = c\left(\frac{h}{p}\right)^2 \|u\|_{1,K} \tag{4.1}$$

where u_I is an interpolant of u. Of course we are not interested in the accuracy of the possible interpolation of the exact solution u, but rather in the accuracy of the finite element approximation of the problem u_h. Still, numerical experiments indicate that the errors given by (4.1) can be considered a rough indication of the accuracy of u_h and can serve as a basis for mesh adaptation.

The major advantage of this method is that it is computationally inexpensive, problem independent, and easy to implement. Yet its rather poor quality is a reason that we use it only if other techniques are not available or if we need only a very rough estimate of the error.

4.1.2 Residual Error Estimate

The idea behind residual *a posteriori* error estimates can be outlined as follows. We substitute the existing finite element approximation u_h into the original statement of the problem being solved. Since u_h is not an exact solution we obtain a certain residual r_h which could be measured in a suitable global norm. For instance, it could be the norm of the space dual to the space containing the solution X:

$$\|r_h\| = \sup_{v \in X} \frac{\langle r_h, v \rangle}{\|v\|} \tag{4.2}$$

for which we are guaranteed that it exists. The exact solution of our original problem is, however, usually unavailable and we can only try to estimate the value of this expression. The techniques leading to such an approximate evaluation of $\|r_h\|$ are referred to as residual error estimates. They express $\|r_h\|$ as a sum of element contributions which we call local error indicators and they also reflect the local accuracy of the solution (i.e., for each element).

The element residual method was originally developed for symmetric elliptic boundary value problems. The method was recently extended to a class of nonsymmetric but symmetrizable problems which includes compressible flow problems [24]. In the following discussion we will provide details for the nonsymmetric version of the method and give only a general outline of the method for the symmetric case. For details about estimating expression (4.2) by local element contributions, we refer to [25]. The presentation below is extracted from [24].

Element Residual Method

Given a domain $\Omega \subset \mathbb{R}^N$ (we assume $N = 2$ for notational simplicity) we consider a general variational boundary value problem in the form

$$\begin{cases} \text{Find } \boldsymbol{u} \in \boldsymbol{X} \text{ such that} \\ B(\boldsymbol{u}, \boldsymbol{v}) = L(\boldsymbol{v}) \text{ for every } \boldsymbol{v} \in \boldsymbol{X} \end{cases} \tag{4.3}$$

where

$$\boldsymbol{X} = \boldsymbol{H}^1(\Omega) = \underbrace{H^1(\Omega) \times \ldots \times H^1(\Omega)}_{n \text{ times}}$$

$$\begin{aligned} B(\boldsymbol{u}, \boldsymbol{v}) &= \sum_{i,j=1}^{n} B_{ij}(u_i, v_j) \\ L(\boldsymbol{v}) &= \sum_{j=1}^{n} L_j(v_j) \end{aligned} \tag{4.4}$$

with the bilinear forms B_{ij} and linear forms L_j defined as (omitting superscripts for notational convenience)

$$\begin{aligned} B(u,v) &= \int_\Omega \left\{ \sum_{k,\ell=1}^{2} a_{kl} \frac{\partial u}{\partial x_\ell} \frac{\partial v}{\partial x_k} + \sum_{k=1}^{2} b_k \frac{\partial u}{\partial x_k} v + \sum_{\ell=1}^{2} d_\ell u \frac{\partial v}{\partial x_\ell} + cuv \right\} dx \\ &+ \int_{\partial\Omega} \left\{ b_s \frac{\partial u}{\partial s} v + d_s u \frac{\partial v}{\partial s} + c_s uv \right\} ds \end{aligned} \tag{4.5}$$

$$\begin{aligned} L(v) &= \int_\Omega \left\{ fv + \sum_{\ell=1}^{2} g_\ell \frac{\partial v}{\partial x_\ell} \right\} dx \\ &+ \int_{\partial\Omega} f_s v \, ds \end{aligned} \tag{4.6}$$

For each pair of indices $i, j = 1, \ldots, n$, a_{kl}, b_k, d_ℓ, c, f, g_ℓ are functions specified in Ω and b_s, d_s, c_s, f_s are functions specified on the boundary $\partial\Omega$. The normal and tangential derivatives on the boundary are defined as

$$\begin{aligned} \frac{\partial u}{\partial n} &= \frac{\partial u}{\partial x_1} n_1 + \frac{\partial u}{\partial x_2} n_2 \\ \frac{\partial u}{\partial s} &= \frac{\partial u}{\partial x_1}(-n_2) + \frac{\partial u}{\partial x_2} n_1 \end{aligned} \tag{4.7}$$

where (n_1, n_2) are components of the outward normal unit vector $\boldsymbol{n}$.

Systems of type (4.3) include not only classical elliptic equations of second-order but also arise naturally as "one time step problems" from different time discretization schemes applied to parabolic or hyperbolic equations. The boundary integrals in (4.5) permit the implementation of different boundary conditions (including Dirichlet boundary conditions via the penalty method).

Replacing $\boldsymbol{X}$ in (4.3) with a finite dimensional subspace $\boldsymbol{X}_{h,p}$ of $\boldsymbol{X}$ we arrive at the approximate problem

$$\begin{cases} \text{Find } \boldsymbol{u}_{h,p} \in \boldsymbol{X}_{h,p} \text{ such that} \\ \\ B(\boldsymbol{u}_{h,p}, \boldsymbol{v}) = L(\boldsymbol{v}) \qquad \forall\, \boldsymbol{v} \in \boldsymbol{X}_{h,p} \end{cases} \tag{4.8}$$

Indices h and p refer here to the use of an arbitrary h-p adaptive finite element (FE) meshes, with locally varying mesh size h and spectral order of approximation p.

It is our goal to propose and investigate here a general method for estimating the *relative residual error* corresponding to (4.8). More precisely, considering the enriched space $\boldsymbol{X}_{h,p+1}$ corresponding to the same mesh but with local order of approximation uniformly increased by one, we define the relative residual error as

$$\sup_{\boldsymbol{v} \in \boldsymbol{X}_{h,p+1}} \frac{|B(\boldsymbol{u}_{h,p}, \boldsymbol{v}) - L(\boldsymbol{v})|}{\|\boldsymbol{v}\|} \tag{4.9}$$

The choice of norm $\|\boldsymbol{v}\|$ is unfortunately not unique. Two important special cases are, however, of interest: the *symmetric case*, when B is symmetric and positive definite, and the *symmetrizable case* when B can be made symmetric by an appropriate change of variables.

Symmetric Case

When the bilinear form B is symmetric and positive definite and the energy norm

$$\|\boldsymbol{v}\|_E = B(\boldsymbol{v}, \boldsymbol{v}) \tag{4.10}$$

is selected in (4.9) the residual error is *equal* to the relative error between $\boldsymbol{u}_{h,p}$ and $\boldsymbol{u}_{h,p+1}$, the FE solution corresponding to the enriched space and measured in the energy norm.

$$\sup_{\boldsymbol{v} \in \boldsymbol{X}_{h,p+1}} \frac{|B(\boldsymbol{u}_{h,p}, \boldsymbol{v}) - L(\boldsymbol{v})|}{\|\boldsymbol{v}\|_E} = \|\boldsymbol{u}_{h,p} - \boldsymbol{u}_{h,p+1}\|_E \tag{4.11}$$

The principal idea behind the proposed error estimate is to interpret (4.11) as a variational formulation of an elliptic problem, transform the bilinear form B into the typical form for elliptic problems, and finally apply the element residual method presented in [25].

Formally, we proceed as follows:

Step 1: Transform formulas (4.5) and (4.6) into the typical form for elliptic equations.

$$\begin{aligned} B(u,v) &= \int_\Omega \left\{ \sum_{k,l=1}^{2} a_{kl} \frac{\partial u}{\partial x_\ell} \frac{\partial v}{\partial x_k} \right. \\ &+ \left. \sum_{k=1}^{2} (b_k - d_k) \frac{\partial u}{\partial x_k} v + \left(c - \sum_{l=1}^{2} \frac{\partial d_l}{\partial x_l} \right) uv \right\} dx \\ &+ \int_{\partial\Omega} \left\{ b_s \frac{\partial u}{\partial s} v + d_s u \frac{\partial v}{\partial s} + \left(c_s + \sum_{\ell=1}^{2} d_l n_l \right) uv \right\} ds \\ L(v) &= \int_\Omega \left(f - \sum_{\ell=1}^{2} \frac{\partial g_\ell}{\partial x_\ell} \right) v dx + \int_{\partial\Omega} \left(f_s + \sum_{\ell=1}^{2} g_\ell n_\ell \right) v ds \end{aligned} \tag{4.12}$$

Step 2: Apply the element residual method to the modified bilinear and linear forms resulting in the estimate

$$\| \boldsymbol{u}_{h,p} - \boldsymbol{u}_{h,p+1} \|_E \leq \left(\sum_K \| \boldsymbol{\varphi}_K \|_{E,K}^2 \right)^{\frac{1}{2}} \tag{4.13}$$

where the *error indicator function* $\boldsymbol{\varphi}_K$ is the solution to the local problem

$$
\left\{
\begin{aligned}
&\text{Find } \varphi_K \in \boldsymbol{X}^0_{h,p+1}(K) \text{ such that} \\
&B_K(\varphi_K, \boldsymbol{v}) = \\
&= \sum_{j=1}^{n} \left\{ \int_K \left\{ f^j - \sum_{i=1}^{2} \frac{\partial g_l^j}{\partial x_l} \right.\right. \\
&\qquad - \sum_{i=1}^{n} \left[- \sum_{k,l=1}^{2} \frac{\partial}{\partial x_k} \left(a_{kl}^{ij} \frac{\partial u_i}{\partial x_l} \right) - \sum_{k=1}^{2} \left(b_k^{ij} - d_k^{ij} \right) \frac{\partial u_i}{\partial x_k} \right. \\
&\qquad\qquad \left.\left. - \left(c^{ij} - \sum_{l=1}^{2} \frac{\partial d_l^{ij}}{\partial x_l} \right) u_i \right] \right\} v_j dx \\
&\qquad + \int_{\partial K \setminus \partial\Omega} \left\{ \left[\!\left[\sum_{i=1}^{2} \sum_{k,l=1}^{2} a_{kl}^{ij} \frac{\partial u_i}{\partial x_l} n_k \right]\!\right] - \sum_{i=1}^{2} \sum_{k,l=1}^{2} a_{kl}^{ij} \frac{\partial u_i}{\partial x_l} n_k \right\} v_j k s \\
&\qquad \int_{\partial K \cup \partial\Omega} \left\{ f_s^j + \sum_{l=1}^{2} g_l^j n_l \right. \\
&\qquad \left. - \sum_{i=1}^{n} \left[b_s^{ij} \frac{\partial u_i}{\partial s} v_j + d_s^{ij} u_i \frac{\partial v_j}{\partial s} + c_s^{ij} u_i v_j \right] \right\} ds \Bigg\} \\
&\text{for every } \boldsymbol{v} \in \boldsymbol{X}^0_{h,p+1}(K)
\end{aligned}
\right. \tag{4.14}
$$

Here $\boldsymbol{X}^0_{h,p+1}(K)$ is the kernel of the h-p interpolation operator defined on the element enriched space $\boldsymbol{X}_{h,p+1}(K)$ or the so-called space of element bubble functions and the element bilinear form B_K is defined as the element contribution to (4.10). Finally, the symbol $[\![\ ]\!]$ denotes the average flux defined along the interelement boundary and evaluated using both the element and the neighboring elements values of derivatives and coefficients a_{kl}^{ij} (if they are discontinuous). The element energy in (4.13) is defined using the element bilinear form B_K.

Step 3: Integrating by parts transforms the element bilinear form and the right-hand side of the local problem into a form consistent with the initial formulas for B and L. Thus, we arrive at the following formulas

$$
\begin{aligned}
B_K(\boldsymbol{\varphi}, \boldsymbol{v}) &= \sum_{i,j=1}^{n} B_K^{ij}(\varphi^i, v_j) \\
L_K(\boldsymbol{v}) &= \sum_{j=1}^{n} L_K^j(v_j)
\end{aligned}
\tag{4.15}
$$

where

$$
\begin{aligned}
B_K(\varphi, v) &= \int_K \left\{ \sum_{k,l=1}^{2} a_{kl} \frac{\partial \varphi}{\partial x_\ell} \frac{\partial v}{\partial x_k} + \sum_{k=1}^{2} b_k \frac{\partial \varphi}{\partial x_k} v + \sum_{\ell=1}^{2} d_\ell \varphi \frac{\partial v}{\partial x_\ell} + c\varphi v \right\} dx \\
&+ \int_{\partial K \setminus \partial\Omega} - \left(\sum_{\ell=1}^{2} g_\ell n_\ell \right) \varphi v \, ds \\
&+ \int_{\partial K \cap \partial\Omega} \left\{ b_s \frac{\partial \varphi}{\partial s} v + d_s \varphi \frac{\partial v}{\partial s} + c_s \varphi v \right\} ds \\
L_K(v) &= \int_K \left(fv + \sum_{\ell=1}^{2} g_\ell \frac{\partial v}{\partial x_\ell} \right) dx \\
&- \int_{\partial K \setminus \partial\Omega} \left\{ \sum_{\ell=1}^{2} g_\ell n_\ell \right) v \, ds \\
&+ \int_{\partial K \cap \partial\Omega} f_s v ds
\end{aligned}
\tag{4.16}
$$

The final form of the local problem is derived as follows:

$$
\begin{cases}
\text{Find } \boldsymbol{\varphi}_K \in \boldsymbol{X}^0_{h,p+1}(K) \text{ such that} \\
B_K(\boldsymbol{\varphi}_K, \boldsymbol{v}) = L_K(\boldsymbol{v}) - B_K(\boldsymbol{u}_{h,p}, \boldsymbol{v}) \\
\qquad + \sum_{i,j=1}^{n} \int_{\partial K \setminus \partial\Omega} \left[\left[\sum_{k,l=1}^{2} a_{kl}^{ij} \frac{\partial u_i}{\partial x_\ell} \right] \right] v_j ds
\end{cases}
\tag{4.17}
$$

Nonsymmetric and Symmetrizable Problems

Formally, formula (4.13) can be used for nonsymmetric problems as well, as long as the local element bilinear forms B_K are positive semidefinite, i.e.,

$$
B_K(\boldsymbol{\varphi}_K, \boldsymbol{\varphi}_K) \geq 0 \tag{4.18}
$$

This happens if the symmetric contributions to B_K dominate the unsymmetric ones (resulting usually from the first–order terms). The global bilinear form B is then automatically semipositive and, with the correct boundary conditions, it is positive definite. This guarantees the well–posedness of the problem.

Another interesting case is when the bilinear form is nonsymmetric but it is *symmetrizable* in the sense that a matrix–valued function $\boldsymbol{A}_0(\boldsymbol{x})$ exists (the so–called symmetrizer introduced earlier) such that a new bilinear form $\tilde{B}$ defined as

$$\tilde{B}(\boldsymbol{u},\boldsymbol{v}) = B(\boldsymbol{u},\boldsymbol{A}_0\boldsymbol{v}) \tag{4.19}$$

is symmetric.

If, in addition, the symmetrized bilinear form $\tilde{B}$ is positive definite, then the error estimation technique can be extended to this case as well.

Introduction of the symmetrizer *does not effect* the construction and solution of the local problems. It only helps identify the norm for the space $\boldsymbol{X}_{h,p+1}$ in (4.9) and affects the evaluation of the error estimate. Using the same definition of element bilinear and linear forms B_K, L_K, we proceed as follows:

Step 1: Use the orthogonality of the residual to the $\boldsymbol{X}_{h,p}$ space,

$$B(\boldsymbol{u}_{h,p},\boldsymbol{v}) - L(\boldsymbol{v}) = B(\boldsymbol{u}_{h,p},\boldsymbol{\psi}) - L(\boldsymbol{\psi}) \tag{4.20}$$

where

$$\boldsymbol{\psi} = \boldsymbol{v} - \Pi_{h,p}\boldsymbol{v} \tag{4.21}$$

where $\Pi_{h,p}$ denotes the h-p interpolation operator (see [25]).

Step 2: Decompose the bilinear and linear forms according to formulas (4.12) introducing the average flux interelement boundary terms

$$\begin{aligned} &B(\boldsymbol{u}_{h,p},\boldsymbol{v}) - L(\boldsymbol{v}) \\ &= \sum_K B_K(\boldsymbol{u}_{h,p},\boldsymbol{\psi}) - L_K(\boldsymbol{\psi}) + \sum_{i,j=1}^{n} \int_{\partial K\setminus\partial\Omega} \left[\!\left[\sum_{k,l=1}^{2} a_{kl}^{ij} \frac{\partial u^i}{\partial x_l} n_k \right]\!\right] \psi^j ds \end{aligned} \tag{4.22}$$

Step 3: Introduce the solutions to the local problems

$$B(\boldsymbol{u}_{h,p},\boldsymbol{v}) - L(\boldsymbol{v}) = \sum_K B_K(\boldsymbol{\varphi}_K,\boldsymbol{\psi}_K) \tag{4.23}$$

where $\boldsymbol{\psi}_K$ is the restriction of $\boldsymbol{\psi}$ to element K.

Step 4: Introduce the symmetrizer and use the Cauchy-Schwartz inequality for the symmetrized form to estimate the error

$$B(\boldsymbol{u}_{h,p},\boldsymbol{v}) - L(\boldsymbol{v}) = \sum_K B_K(\boldsymbol{\varphi}_K, \boldsymbol{A}_0\boldsymbol{A}_0^{-1}\boldsymbol{\psi}_K)$$

$$= \sum_K \tilde{B}_K(\boldsymbol{\varphi}_K, \boldsymbol{A}_0^{-1}\boldsymbol{\psi}_K) \leq \sum_K \tilde{B}(\boldsymbol{\varphi}_K,\boldsymbol{\varphi}_K)^{\frac{1}{2}} \tilde{B}_K(\boldsymbol{A}_0^{-1}\boldsymbol{\psi}_K, \boldsymbol{A}_0^{-1}\boldsymbol{\psi}_K)^{\frac{1}{2}} \tag{4.24}$$

$$\leq C \left[\sum_K B_K(\boldsymbol{\varphi}_K, \boldsymbol{A}_0\boldsymbol{\varphi}_K)\right]^{\frac{1}{2}} B(\boldsymbol{A}_0^{-1}\boldsymbol{v},\boldsymbol{v})^{\frac{1}{2}}$$

Here $C = \max_K C_K$ where for every element K, C_K is identified as the norm of $(I - \Pi_{h,p})$ operator with respect to the element energy norm defined as

$$\|\boldsymbol{v}\|_{E,K}^2 = B_K(\boldsymbol{A}_0^{-1}\boldsymbol{v},\boldsymbol{v}) \tag{4.25}$$

(see [25] for a detailed discussion of C). For undistorted meshes C is close to one (independent of the order of approximation) and in practical calculations is neglected.

Identifying the global energy norm for $\boldsymbol{v}$ in (4.9) as the sum of (4.26) we arrive at the final estimate of the form

$$\sup_{\boldsymbol{v}\in\boldsymbol{X}_{h,p+1}} \frac{|B(\boldsymbol{u}_{h,p},\boldsymbol{v}) - L(\boldsymbol{v})|}{\|\boldsymbol{v}\|} \leq \left[\sum_K B_K(\boldsymbol{\varphi}_K, \boldsymbol{A}_0\boldsymbol{\varphi}_K)\right]^{\frac{1}{2}} \tag{4.26}$$

Example: *Taylor-Galerkin Method for Euler Equations*

Recall that the variational formulation of the Taylor-Galerkin method is of the general form (4.3) with the bilinear form defined as follows.

$$B\left(\boldsymbol{U}^{n+1},\boldsymbol{V}\right) = \int_\Omega \left\{\boldsymbol{V}^T\boldsymbol{U}^{n+1} + \frac{\Delta t^2}{2}\sum_{k,l=1}^{2}\left(\frac{\partial \boldsymbol{V}}{\partial x_k}\right)^T \boldsymbol{A}_k\boldsymbol{A}_l\frac{\partial \boldsymbol{U}^{n+1}}{\partial x_\ell}\right\} dx$$

$$+ \quad \text{boundary terms} \tag{4.27}$$

The form of boundary terms present in the formula for the bilinear form depends on boundary conditions.

The formulation is *nonsymmetric.* However, it is known (see [16,14]) that there exists a symmetrizer $\boldsymbol{A}_0 = \boldsymbol{A}_0(\boldsymbol{u})$, see Fig. 2.1 (Hessian of the entropy function for Euler equations),

such that

$$\begin{aligned} &1.\quad \boldsymbol{A}_0 = \boldsymbol{A}_0^T > 0 \\ &2.\quad (\boldsymbol{A}_0\boldsymbol{A}_i)^T = \boldsymbol{A}_0\boldsymbol{A}_i \geq 0 \end{aligned} \tag{4.28}$$

Based on (4.28), one can easily verify that (with a proper treatment of boundary conditions) the bilinear form

$$\tilde{B}(\boldsymbol{U},\boldsymbol{V}) = B(\boldsymbol{U},\boldsymbol{A}_0\boldsymbol{V}) \tag{4.29}$$

is symmetric, provided the derivatives of the symmetrizer $\boldsymbol{A}_0$ are negligible, i.e.,

$$\frac{\partial}{\partial x_i}\boldsymbol{A}_0 \approx \boldsymbol{0} \tag{4.30}$$

Example: *The Momentum Step of the Navier-Stokes Equations*

The momentum step in the two-step procedure outlined in Section 2 involves solving the system of equations:

$$m_j^{n+1} - \beta\Delta t\sum_{i=1}^{2}\tau_{ij,i}^{n+1} = m_j^n + (1-\beta)\Delta t\sum_{i=1}^{1}\tau_{ij,i}^n \tag{4.31}$$

Equations (4.31), if rewritten in terms of the velocity components, reduces to a system of two symmetric, elliptic equations. Unfortunately, in order to comply with the conservative form of the equations, (4.31) *must be solved* in momentum components.

The variational formulation of (4.31) does not result in a symmetric problem but the bilinear form may be symmetrized using the symmetrizer

$$\boldsymbol{A}_0 = \begin{bmatrix} \dfrac{1}{\rho} & 0 \\ 0 & \dfrac{1}{\rho} \end{bmatrix}$$

as this transforms the problem to a symmetric formulation in velocity components.

Example: *The Energy Step for Navier-Stokes Equations*

The energy step involves solving the equations:

$$\begin{aligned} &e^{n+1} - \beta\Delta t\sum_{i=1}^{2}\left(\sum_{j=1}^{2}\tau_{ij}^{n+1}u_j^{n+1} + \kappa T_{,i}^{n+1}\right)_{,i} \\ &= e^n + (1-\beta)\Delta t\sum_{i=1}^{2}\left(\sum_{j=1}^{2}\tau_{ij}^{n}u_j^{n} + \kappa T_{,i}^{n}\right)_{,i} \end{aligned} \tag{4.32}$$

The variational formulation of (4.32) is not symmetric. However, since rewriting (4.32) in terms of temperature T results in a symmetric diffusion equation, and since $e = c_v \rho T + m^2/2\rho$, the factor $A_0 = 1/\rho$ is a suitable symmetrizer of the problem.

The extension of the element residual error estimation to the implicit *slash* explicit method (which is based on one-step Taylor-Galerkin Formulation) is straightforward: first we calculate the error indicator function by solving the local problem (4.23) for each element, then use (4.26) to compute the error indicator for that element. The bilinear form is obtained from the variational formulation of the problem as before, and is of the general form (3). The same symmetrizer used for Euler equation can also be applied for Navier-Stokes equations (cf Hughes' paper). It should be noted that, although the algorithm extends in a neutral way, the theoretical work for the Navier-Stokes equation is still not complete.

Numerical Examples

In this section, two example problems illustrating these techniques of error estimation are presented. Note, that these are rather simple examples designed to illustrate the basic ideas presented here on relatively worse meshes. More practical applications we presented in Section 7. The results take the form of plots of the error estimates and effectivity indices as well as global effectivity indices and standard deviations. These quantities are defined as follows:

$$\gamma_K = \frac{\theta_K}{|||e|||_K} \tag{4.33}$$

where γ_K is the effectivity index for element K, θ_K is the estimated error and $|||e|||_K$ is the actual element error in the coarse mesh approximation (comparing the coarse mesh approximation with either the analytic solution or the approximate solution on a mesh of uniformly increased polynomial order). Additionally, we introduce a discrete measure (weight) ω_K defined according to

$$\omega_K = \frac{|||e|||_K^2}{|||e|||^2} \tag{4.34}$$

With this definition, the global effectivity index becomes:

$$\gamma^2 = \frac{\theta^2}{|||e|||^2} = \frac{\left(\sum_K \theta_K^2\right)}{|||e|||^2} = \sum_K \gamma_K^2 \omega_K \tag{4.35}$$

Now classical statistics suggest a standard deviation σ (with respect to the measure) as a method to quantify the ability of the estimates to predict an appropriate distribution of error. The standard deviation is defined as:

$$\sigma^2 = \sum_K \left(\gamma_K^2 - \gamma^2\right)^2 \omega_K \tag{4.36}$$

In order to eliminate any global constants that may be missing from our estimates, we normalize the element effectivity indices by dividing them by the global effectivity index:

$$\overline{\gamma}_K = \frac{\theta_K}{|||e|||_K} \cdot \gamma^{-1} \tag{4.37}$$

which results in a standard deviation defined according to:

$$\overline{\sigma}^2 = \sum_K \left(\overline{\gamma}_K^2 - 1\right)^2 \omega_K \tag{4.38}$$

Example 1: Inviscid Flow Over a Blunt Body

We used the Taylor-Galerkin method described in Section 2 to model the flow over a blunt body with Mach number $M = 6$. Figure 4.1 shows the density contours of a steady-state solution obtained on a uniform mesh of 16×16 linear elements. Figures 4.2a and 4.2b present distributions of the error indicators θ_K (obtained using (4.26)) and the normalized effectivity indices $\overline{\gamma}_K$ (4.37). Since the exact solution to the problem is not available, the exact errors are not known. For this reason we computed the effectivity indices $\gamma_K = \theta_K / |||e|||_K$, using instead of the true errors $|||e|||_K$, the errors understood as a difference between the actual finite element solution and the solution obtained by performing one time step on the mesh enriched to quadratic elements. It can be observed that the error indicator correctly picked up the shock as the maximum error region. It is important to note that figure 4.2.6 presents effectivity index $\tilde{\gamma}_K$, not the error indicator. Due to the presence of the value of error in the denominator of the definition of $\tilde{\gamma}_K$ (4.37), the effectivity index will often exhibit overshoots in the areas of low error (division by small numbers). That explains presence of high values of effectivity index in front of the low shock or in front of the plate in the next example. The global effectivity index for this problem was $\gamma = 7.7$ and a standard deviation of local effectivity indices $\overline{\sigma} = 1.67$.

Example 2: Viscous Flow Over a Flat Plate

The two–step algorithm was also used to model the viscous flow past a flat plate. The problem being modeled was designed by the following data:

- Mach number, $M = 3$
- Reynolds number, $Re = 500$
- Free stream temperature $T_\infty = 80°K$
- The temperature of the plate, $T_w = 228°K$

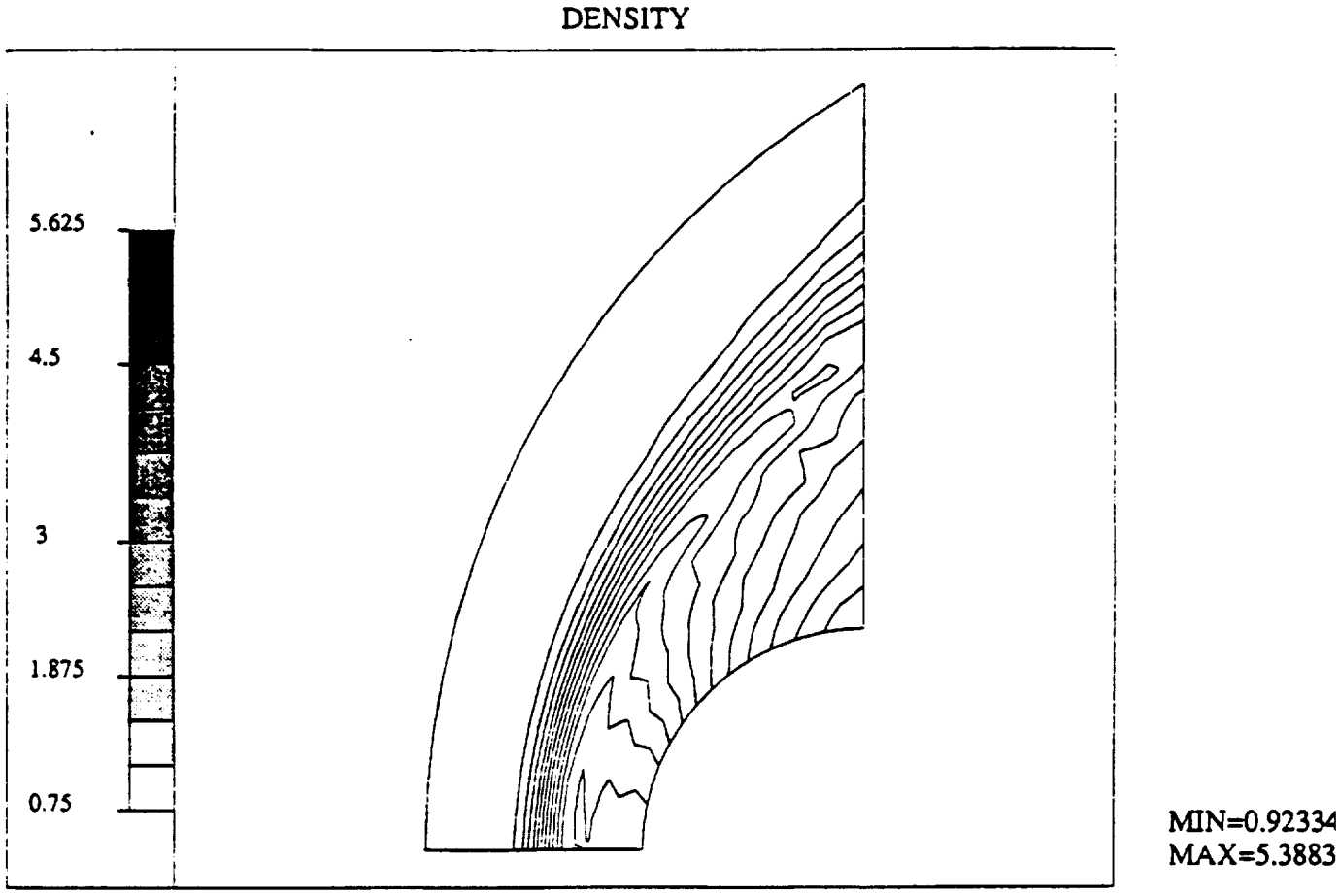


Figure 4.1: Flow over a blunt body, $M = 6$. Density contours.

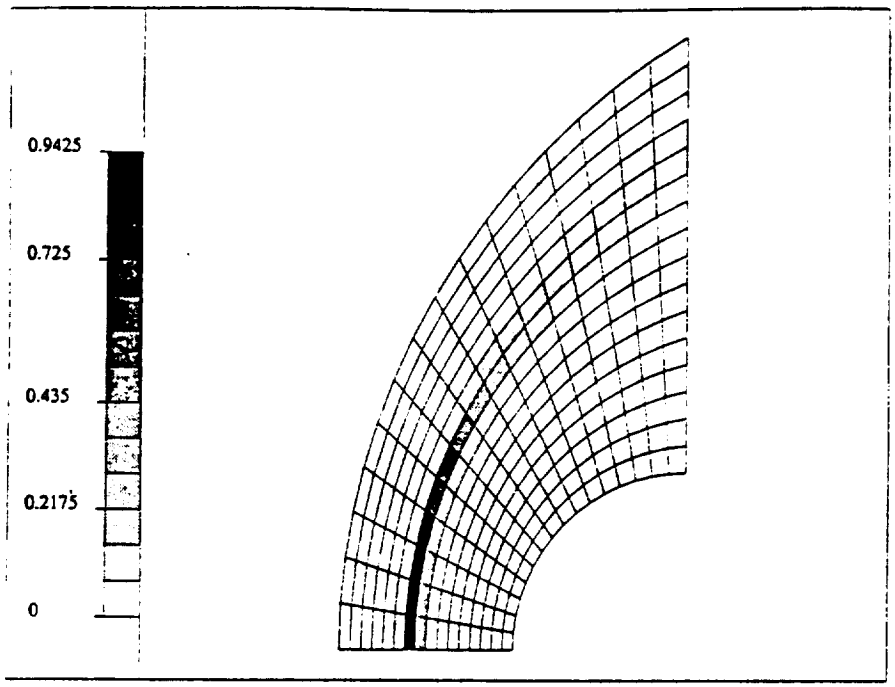


Figure 4.2: (a) Flow over a blunt body. Distribution of error indicators.

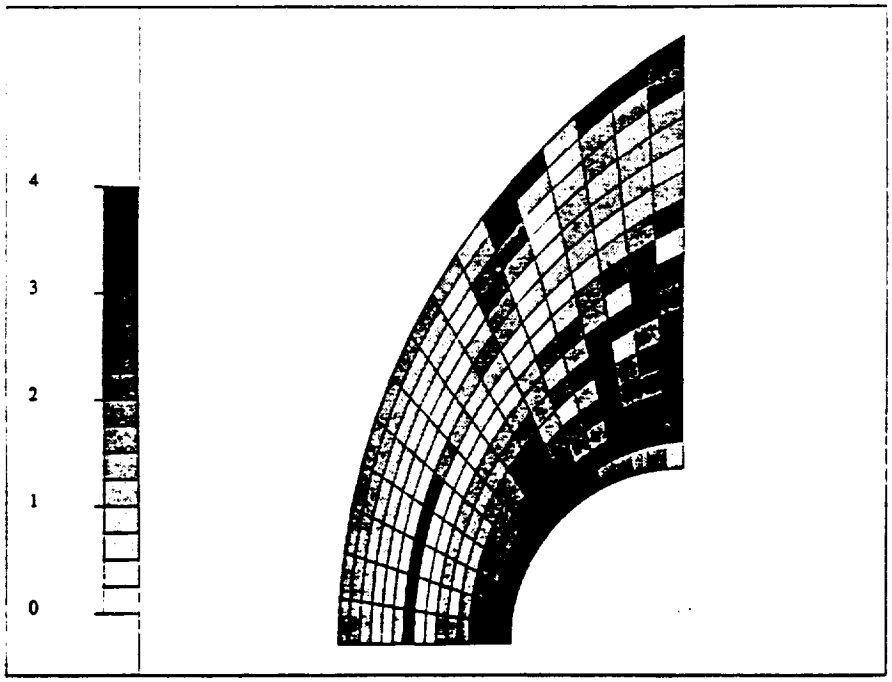


Figure 4.2: (b) Flow over a blunt body. Local effectivity indices.

The finite element mesh is shown in Fig. 4.3. We applied initial h and p refinements to introduce appropriate layers of small higher order (up to $p = 3$) elements along the plate to resolve the boundary layer phenomena. Different shades of gray in Fig. 4.3 correspond to different orders of approximation. Elements with only their sides shaded are anisotropic elements with a higher order approximation in the direction perpendicular to the plate only. The solution of the flat plate problem in terms of density contours is presented in Fig. 4.4.

Since the two–step algorithm consists of three linear steps, we performed an error estimation for each step. Similarly, as in Example 1, the exact errors involved in effectivity indices analysis were replaced by the errors obtained as differences between the actual solutions of Euler, momentum and energy steps, and the corresponding solutions obtained by enriching the order of approximation by 1 throughout the mesh, and performing one Euler or momentum, or energy time step, respectively. These differences were then measured in the energy norms defined by the bilinear forms associated with these steps, symmetrized as described in previous sections.

Figures 4.5, 4.6, and 4.7 present distributions of the error indicators and local effectivity indices for the three steps of the two–step algorithm. The global effectivity indices γ and standard deviations of local effectivity indices, $\overline{\sigma}$, in this problem were as follows:

Euler step	$\gamma = 18.7$	$\overline{\sigma} = 6.2$
momentum step	$\gamma = 25.9$	$\overline{\sigma} = 5.8$
energy step	$\gamma = 3.8$	$\overline{\sigma} = 7.4$

4.1.3 Relative Error Estimate

The idea of the relative error estimate is to compare the finite element solution on a current mesh with a solution obtained on an enriched mesh and to measure the difference between the two solutions in a suitable norm. The enrichment of the mesh is done by raising the order of approximation of all elements by one. Of course, solving the problem on the enriched mesh is much more expensive than obtaining the original solution, so the method apparently does not seem very reasonable. However, if the original solution is a result of some expensive iterative process (such as, for instance, converging to a steady state solution in the case of viscous flow problems), then performing a single extra linear step on an enriched mesh is not a significant part of the total cost of the computations. In addition, solving of this problem can be performed with an iterative equation solver with a very good initial guess and with a very limited number of iterations (limited even to just one iteration).

As a norm measuring the difference between the two solutions, one can use the energy

C-2

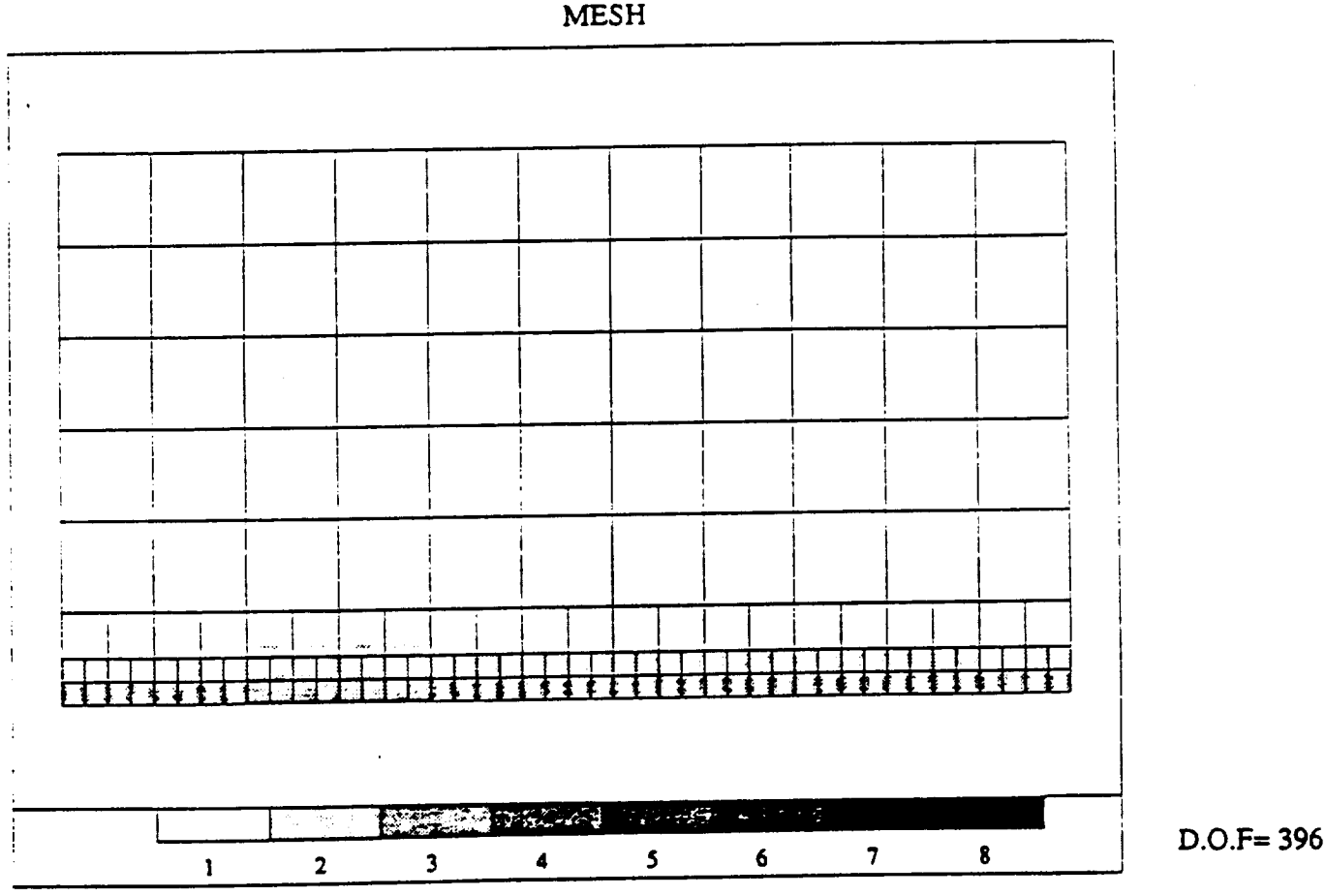


Figure 4.3: Flat plate problem. An *h-p* finite element mesh.

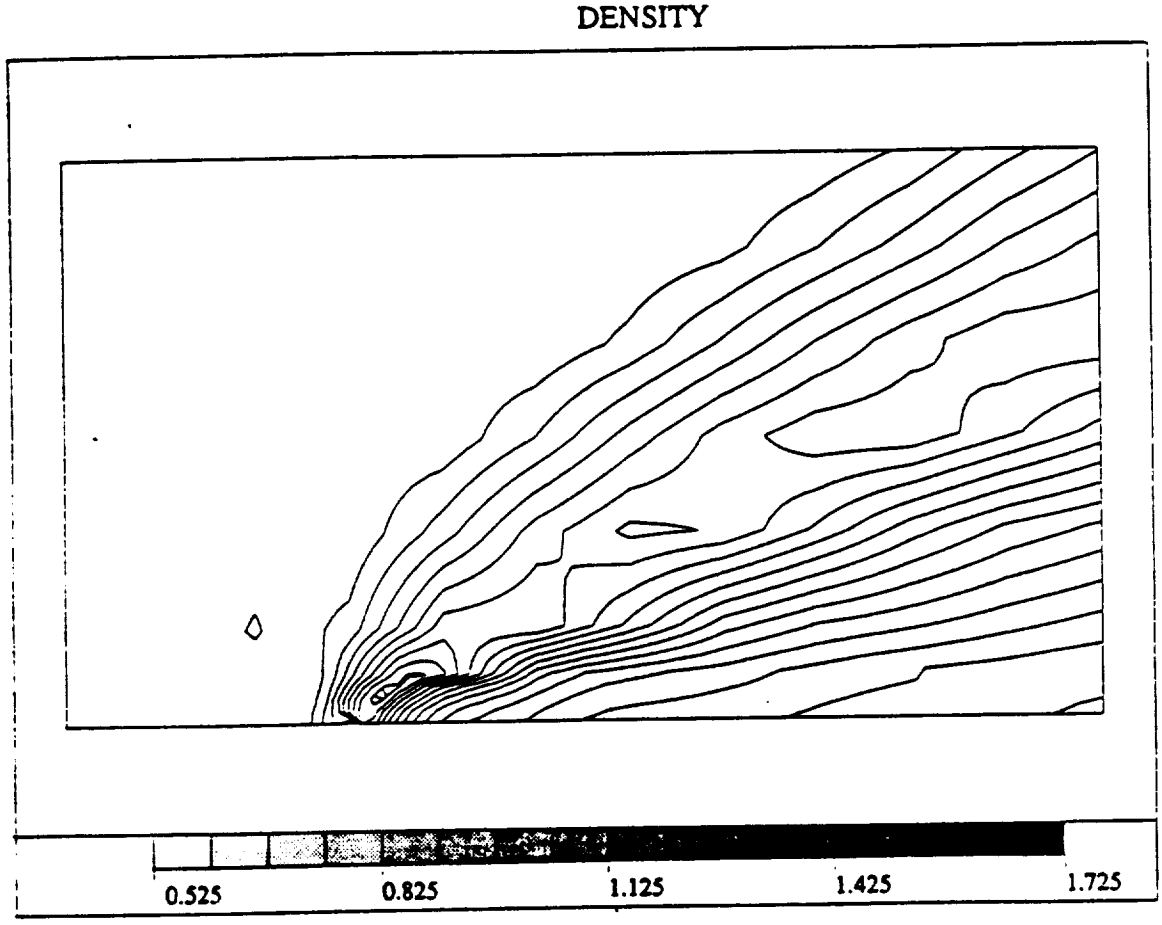
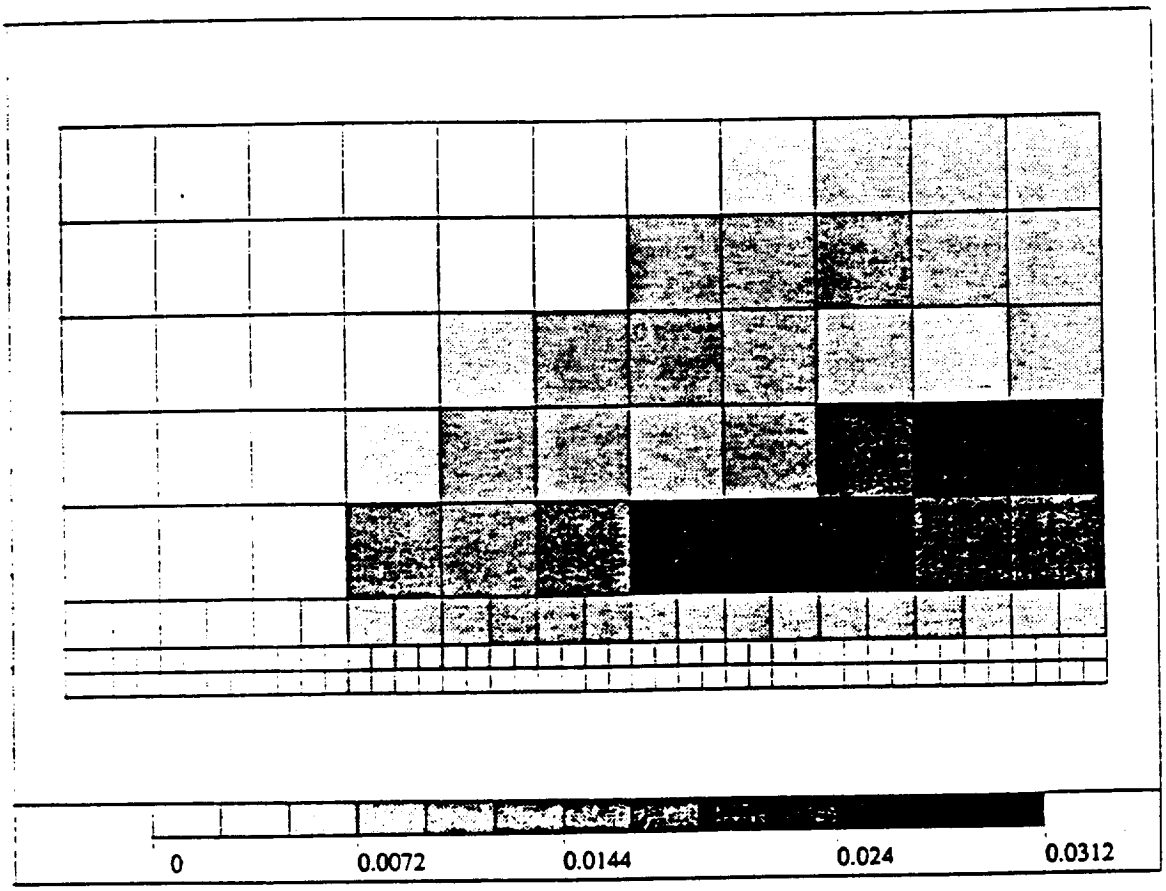


Figure 4.4: Flat plate problem. Density contours.



MIN= 0.389E-05
MAX=0.0302722
ERROR=0.0961987
D.O.F= 958

Figure 4.5: (a) Flat plate problem. Error indicators for the Euler step.

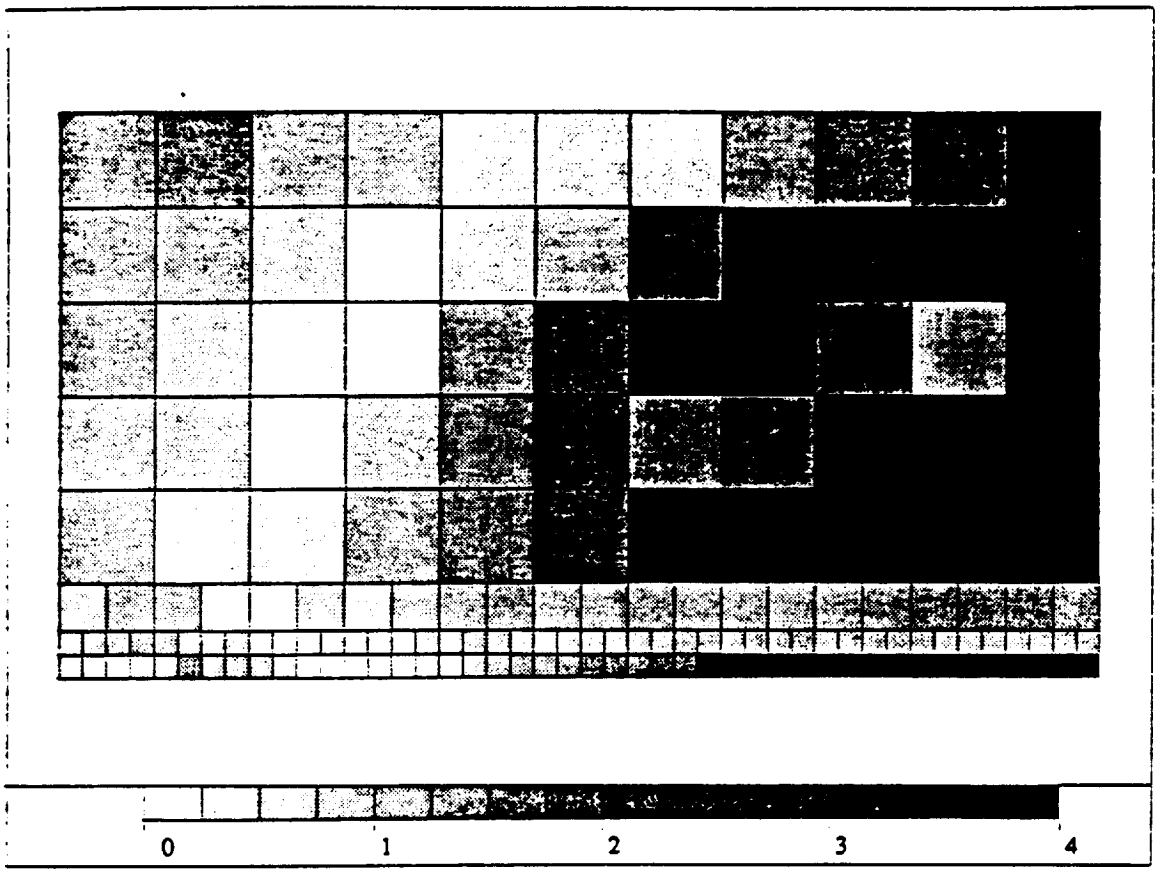
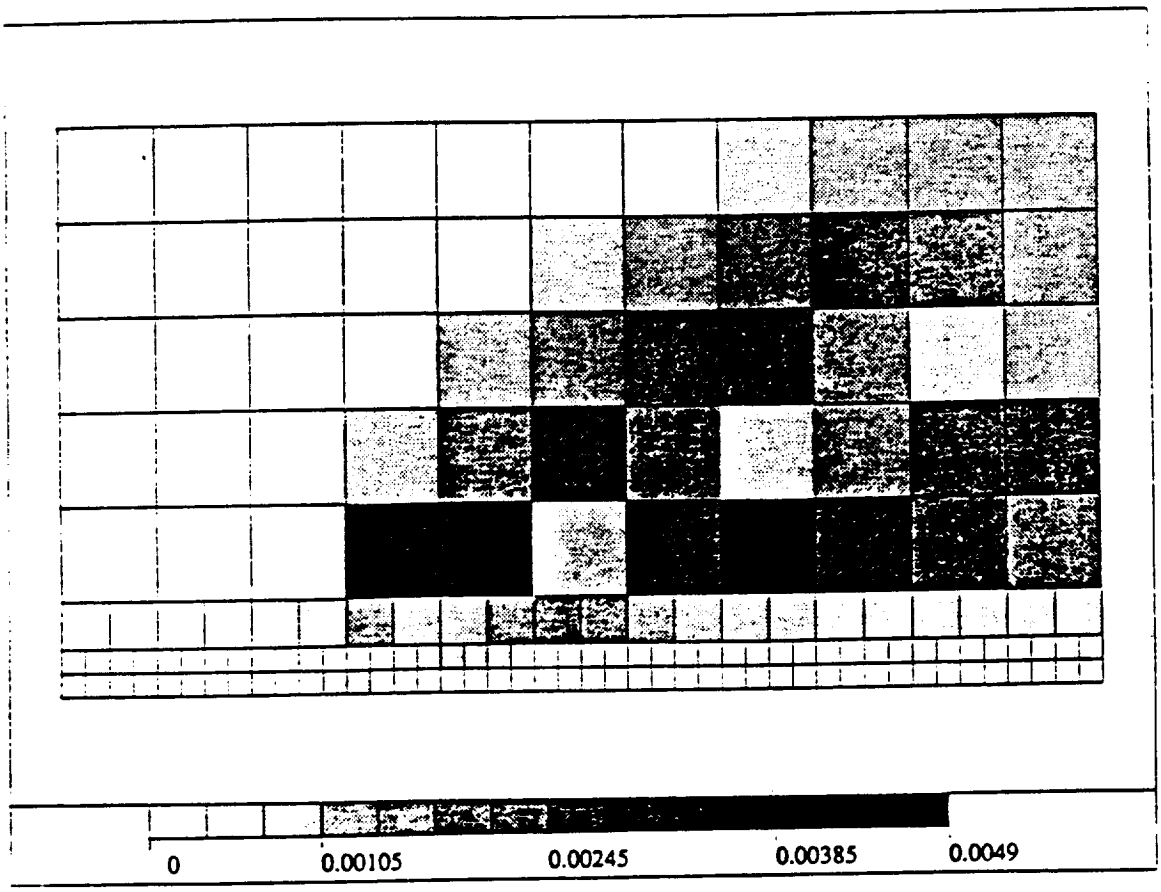


Figure 4.5: (b) Flat plate problem. Local effectivity indices for the Euler step.



MIN= 0.183E-05
MAX=0.0045631
ERROR=0.0154444
D.O.F= 396

Figure 4.6: (a) Flat plate problem. Error indicators for the momentum step.

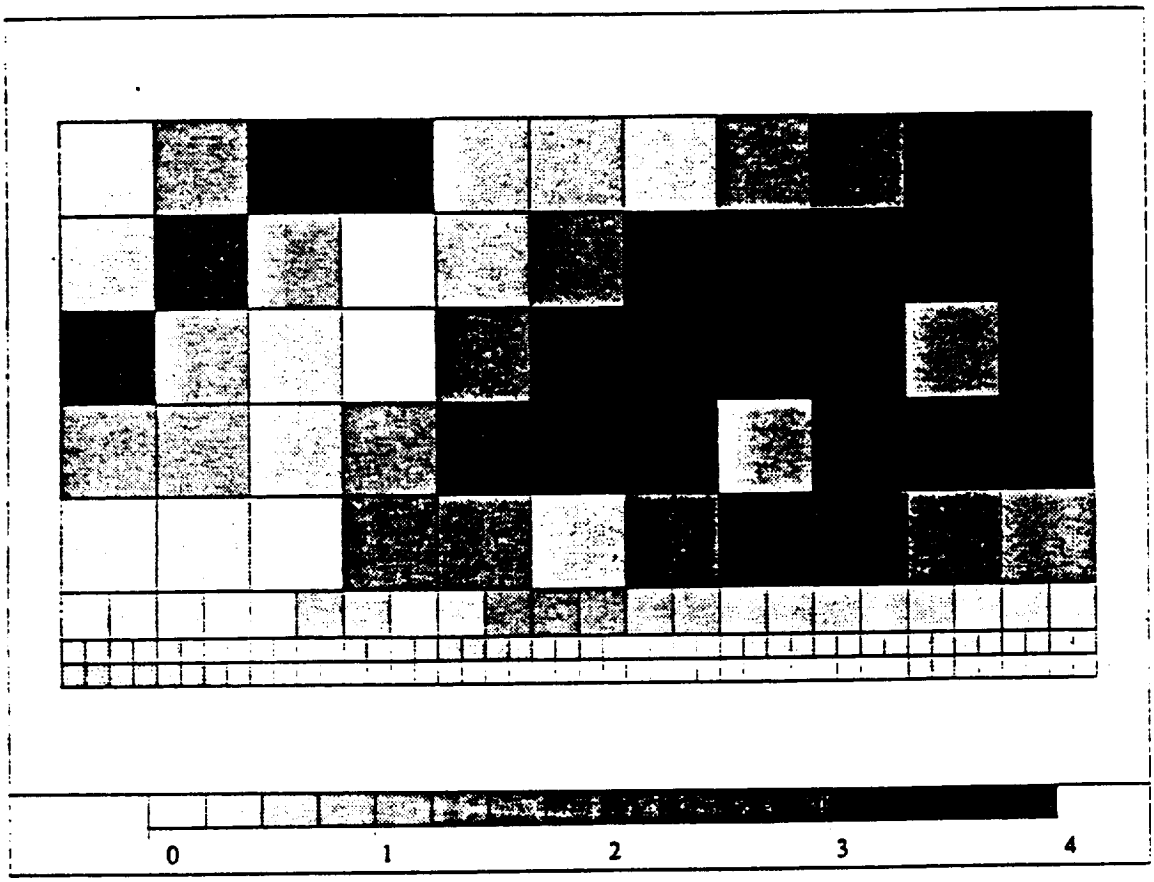
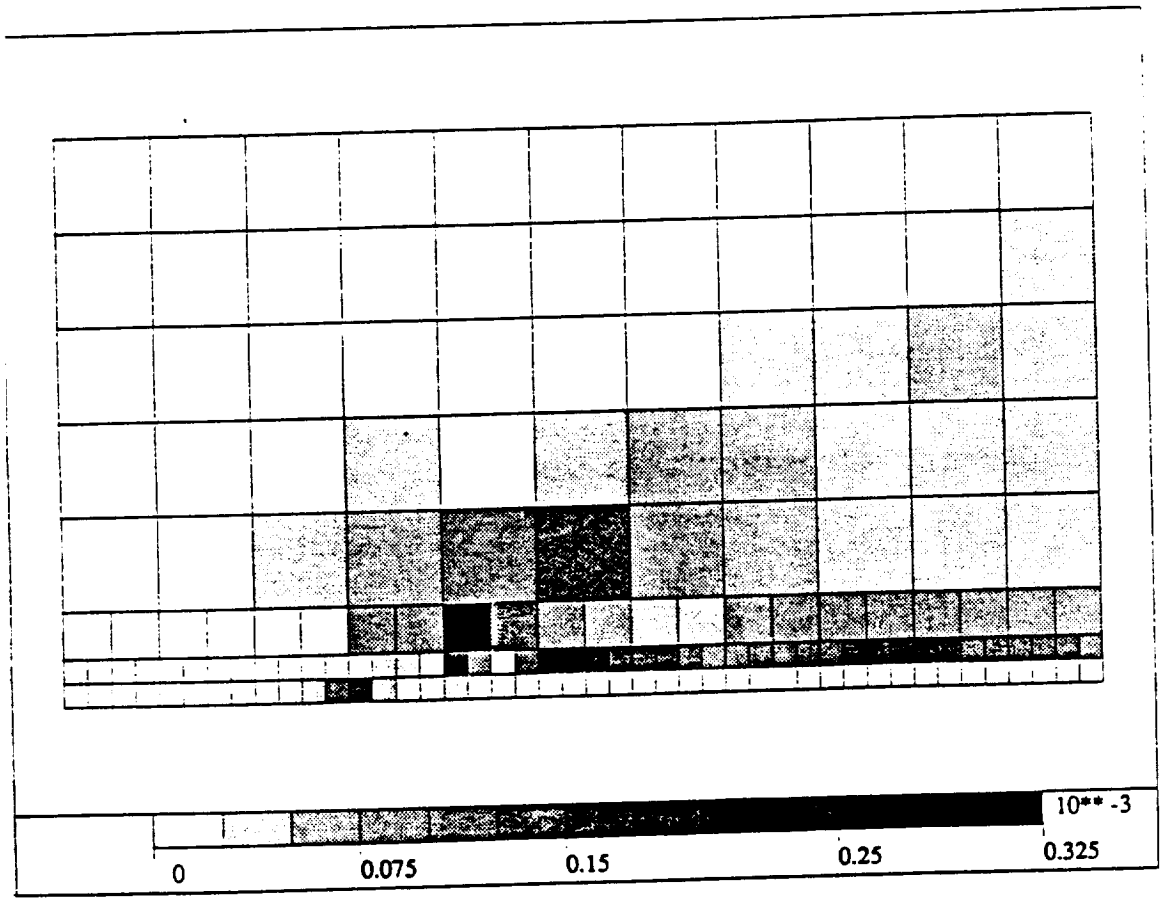


Figure 4.6: (b) Flat plate problem. Local effectivity indices for the momentum step.



MIN= 0.212E-06
MAX= 0.316E-03
ERROR=0.0010473
D.O.F= 958

Figure 4.7: (a) Flat plate problem. Error indicators for the energy step.

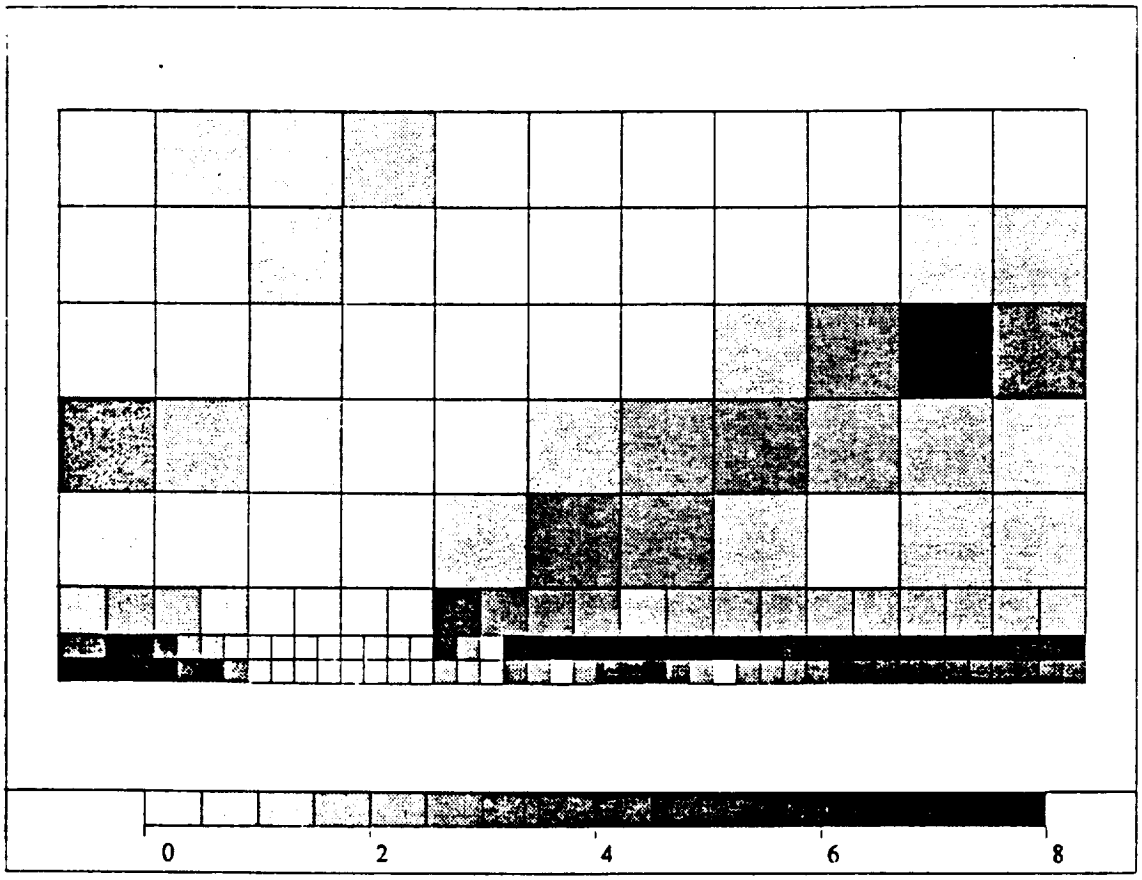


Figure 4.7: (b) Flat plate problem. Local effectivity indices for the energy step.

norm in the case of symmetric problems, or the norm defined by the symmetrized bilinear form of the original problem in case it is nonsymmetric but symmetrizable. With such choices of norms, the element residual method discussed in the previous section is an approximation of the relative error estimate. In fact, the element residual method approximates errors defined by the relative error estimate by expressing them in the form of local element contributions which are evaluated without actually solving the problem on an enriched mesh.

4.2 Directional Adaptation Indicator

The error indicators calculated from the element residual method have been used successfully for the h-refinement for a number of hypersonic inviscid and viscous problems. During the last year of this project, we have also implemented directionally-dependent error estimate schemes applicable to the h-p compressible flow solver. These directional adaptation indicators will be discussed in this section. The current h-p data structure allows two kinds of mesh adaptation: h-refinement (refine/unrefine elements) and p-enrichment (isotropically/ansitropically increase the spectral orders of elements). Although the present h-p data structure only allows directional p-enrichment, the methodology discussed here is applicable to both directional h-refinement and p-enrichment in two- and three-dimensional problems.

It should be noted here that, in general, there exist no formal definition of directional error estimate – error norms used in the adaption process are defined in a full three-dimensional or two-dimensional spaces. The goal of our research is to provide directional *adaptation* indicator, which can choose an optimal refinement/enrichment direction. By optimal we understand a direction which provides maximum reduction of error norm due to a directional refinement/enrichment.

According to this definition, the most natural way of defining directional adaptation indicator would consist of the following steps:

1. try to refine/enrich the element in each of master directions (two or three depending on problem dimensionality),
2. for each trial direction, estimate the error after the refinement,
3. choose the adaptation direction which provides greatest error reduction.

The above method, although formally correct, would be computationally too expensive. For practical purposes, we adopt two approaches which provide the directional adaptation indicator as a relatively simple and inexpensive extension of basic error estimation procedures. Construction of such an indicator is presented below. For the sake of clarity, we focus on a two-dimensional case. Extensions to three dimensions are immediate.

The first approach is based on the element residual method. Recall that the error indicator function φ_K is computed based on the element enriched space, $\boldsymbol{X}_{h,p+1}(K)$. This function can be effectively used as an indicator to determine the directionality of the error for that element. A natural choice is to use the norm of the directional derivative of the error indicator function in each coordinate on the master element

$$||\varphi_{,\xi}||^2_{0,K} = \int_{\Omega_K} (\frac{\partial \varphi_K}{\partial \xi})^2 d\boldsymbol{x} \qquad \text{and} \qquad ||\varphi_{,\eta}||^2_{0,K} = \int_{\Omega_K} (\frac{\partial \varphi_K}{\partial \eta})^2 d\boldsymbol{x} \tag{4.39}$$

as the directional indicator. The actual refinement/enrichment direction is that of maximum norm of derivative of error function. This procedure is rather intuitive and theoretically unexplored, however, it has received a consistent support among researchers in the area of error estimation. The effectiveness of this directional adaptation indicator can only be confirmed by numerical experiment.

The second approach is based more on interpolation error estimator. In particular, it focuses on different contributing components of the semi-norm of the solution:

$$|U|^2_{1,K} = ||U_{,\xi}||^2_{0,K} + ||U_{,\eta}||^2_{0,K}$$

For practical purposes, the exact solution $\boldsymbol{u}$ can be replaced with the finite element solution $\boldsymbol{U}$. Then, to determine the possible enrichment directions to improve the quality of the solution, one can utilize the norms of directional derivatives of the solution

$$||U_{,\xi}||^2_{0,K} = \int_{\Omega_K} (\frac{\partial U}{\partial \xi})^2 d\boldsymbol{x} \qquad \text{and} \qquad ||U_{,\eta}||^2_{0,K} = \int_{\Omega_K} (\frac{\partial U}{\partial \eta})^2 d\boldsymbol{x} \tag{4.40}$$

Note that these values are only the local properties – they represent, for each element, the directional variations of either the error function or the solution. By selecting one of these norms and normalizing the $\xi\eta$-derivatives with respect to the sum of the two derivatives, a directional adaptation indicator can be defined as

$$\phi_{K\eta} = \frac{\int_{\Omega_K} (\frac{\partial \varphi_K}{\partial \eta})^2 d\boldsymbol{x}}{\int_{\Omega_K} \left[(\frac{\partial \varphi_K}{\partial \xi})^2 d\boldsymbol{x} + (\frac{\partial \varphi_K}{\partial \eta})^2 \right] d\boldsymbol{x}} \tag{4.41}$$

or

$$\phi_{K\eta} = \frac{\int_{\Omega_K} (\frac{\partial U}{\partial \eta})^2 d\boldsymbol{x}}{\int_{\Omega_K} \left[(\frac{\partial U}{\partial \xi})^2 + (\frac{\partial U}{\partial \eta})^2 \right] d\boldsymbol{x}} \tag{4.42}$$

The normalization gives $\phi_{K\eta}$ a value between 0 and 1, with the "indication" of enrichment in ξ- or η-direction according to whether the values is close to 0 or 1, respectively.

In practice, we first use the element residual method presented in the previous section to determine whether an element needs to be enriched. If the residual error for an element exceeds the user specified threshold value, we then compute the direction adaptation indicator

defined above. According the user selected values for b_1 and b_2, where $0 \leq b_1 < b_2 \leq 1$, the element is enriched as follows:

- anisotropically enrich in ξ-direction if $0 \leq \phi_{K\eta} < b_1$
- anisotropically enrich in η-direction if $b_2 < \phi_{K\eta} \leq 1$
- isotropically enrich in both directions if $b_1 \leq \phi_{K\eta} \leq b_2$

The values of b_1 and b_2, which control the directionality of the p-enrichment, are currently being selected based on numerical experience with typical values ranging between 0.2 through 0.4 for b_1 and 0.6 through 0.8 for b_2.

Numerical Example: Carter's Flat Plate Problem

The directional adaptation indicator described above has been applied to the Carter's flat plate problem. For this problem we have converged the solution on an initial linear graded mesh with two levels of h-refinements as shown in Fig. 4.8. The corresponding density contours are also shown in Fig. 4.9. A map of the error indicator, θ_K, calculated by the element residual method (as presented in Sec. 4.1) is shown in Fig. 4.10. Note, only the elements with error indicator values, θ_K, greater than the threshold value (10^{-5} for this example) are considered for enrichment. Fig. 4.11 shows the corresponding plot of the directional adaptation indicator, $\phi_{K\eta}$, calculated from the directional derivatives of the error indicator function in equation (4.41). Note that the elements with θ_K less than 10^{-5} (the ones not to be enriched) are not shaded in the plot. The next two figures, 4.12 and 4.13, present the resulting meshes after one p-enrichment pass for the values of (b_1, b_2) set to $(0.2, 0.8)$ and $(0.3, 0.7)$, respectively.

Similar estimations were performed using the directional adaption indicators based on solution gradients. The corresponding plots of directionality indicator $\varphi_{K\eta}$ and the corresponding enriched meshes are shown in figures 4.14 to 4.16, respectively.

A careful study of the above results clearly shows that both proposed directional indicators perform a good job of suggesting directional p-enrichment: isotropic at the plate tip region and normal to the wall within the developed boundary layer. It seems that directional indicators based on the interpolation error estimate (solution gradients) behaves in a more consistent fashion than directional indicator, based on residual error estimate. This is probably because the solution of the local residual error estimation problem is more sensitive to element size and boundary conditions than the actual solution of the problem.

Note that the well-known low effectivity of interpolation error estimators is not a problem in this case, because we are using the directionality indicator to choose between different enrichment directions, and not to estimate the actual directional error.

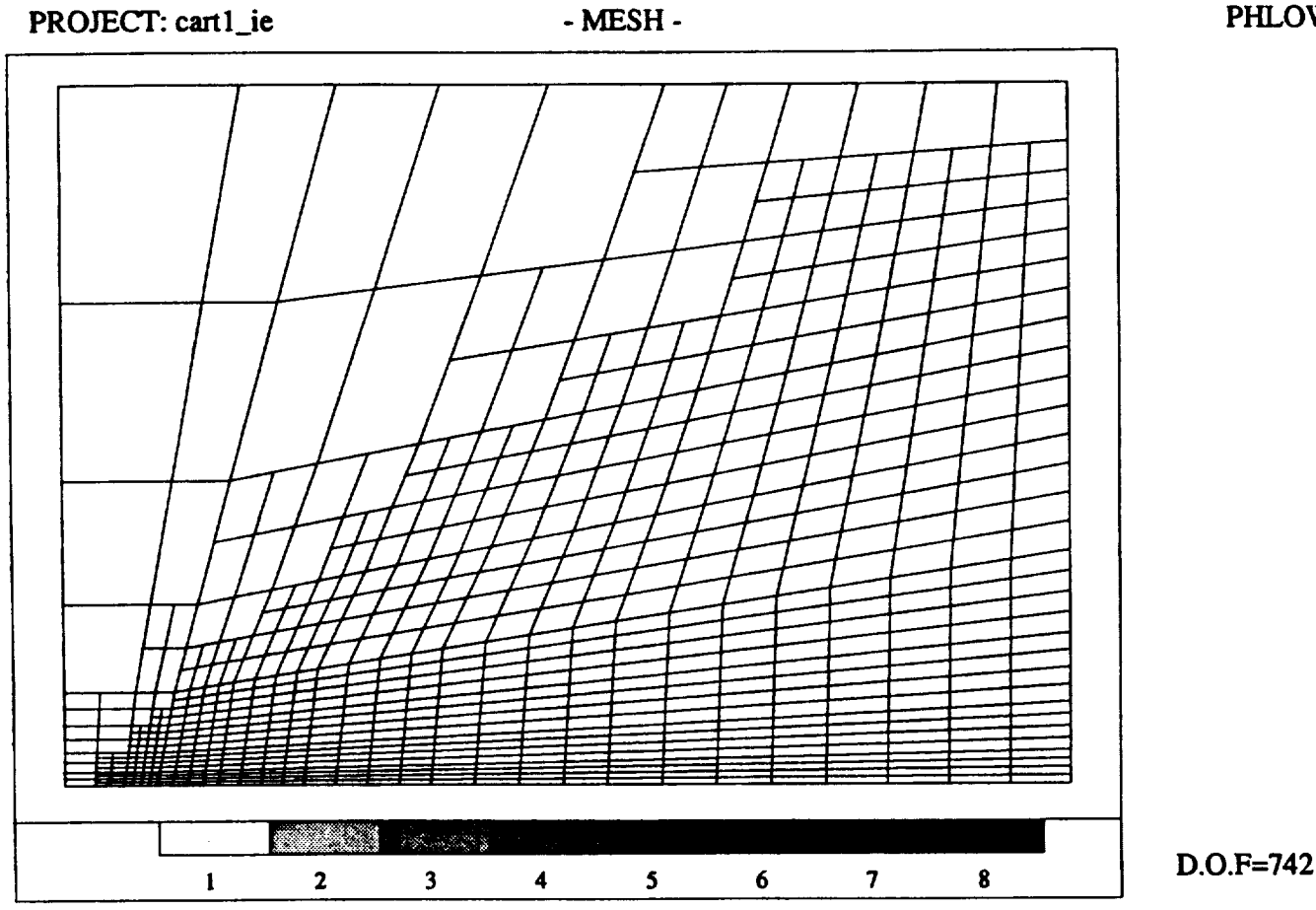


Figure 4.8: Carter's flat plate problem: Mesh with second level refinement.

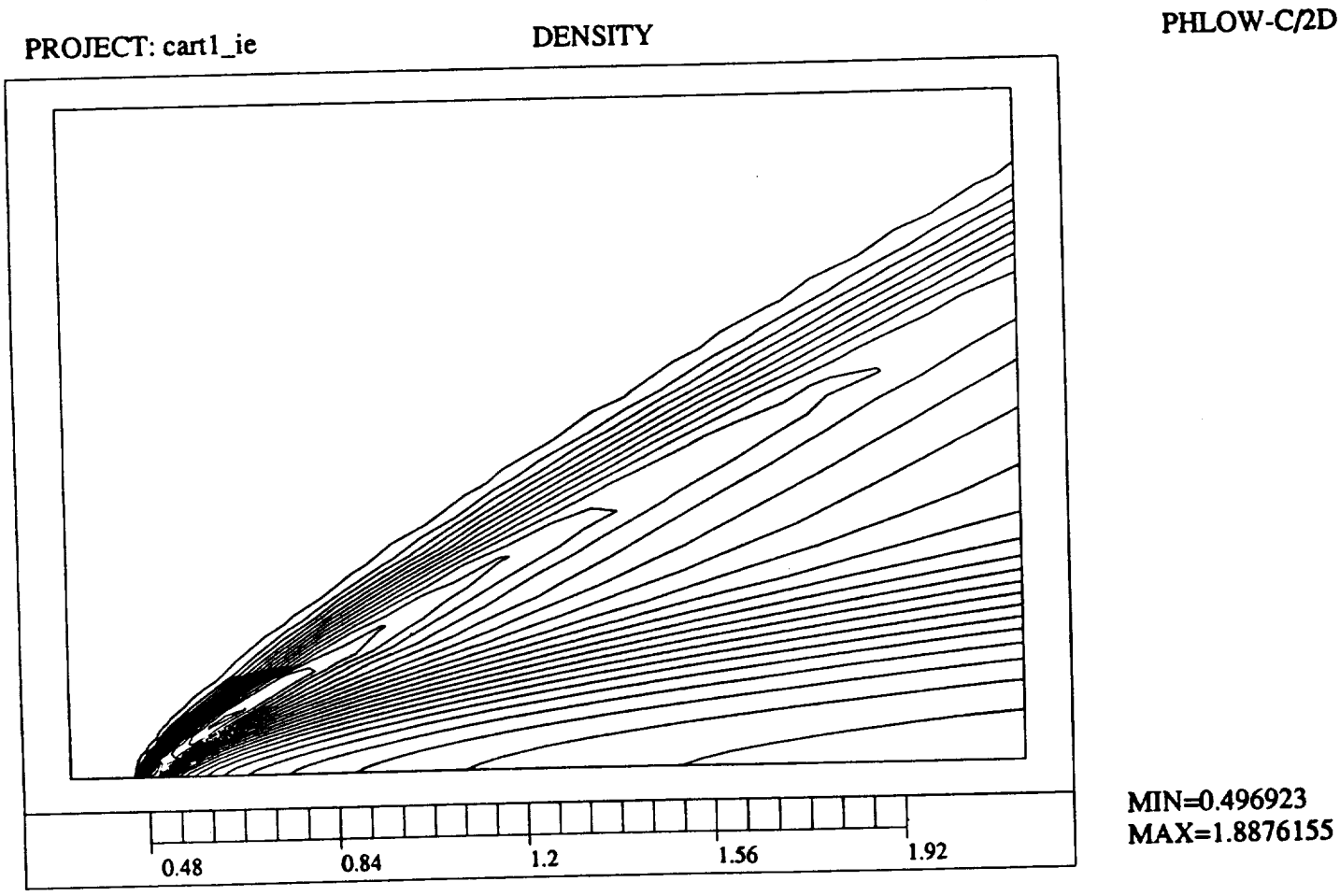


Figure 4.9: Carter's problem: Density contours.

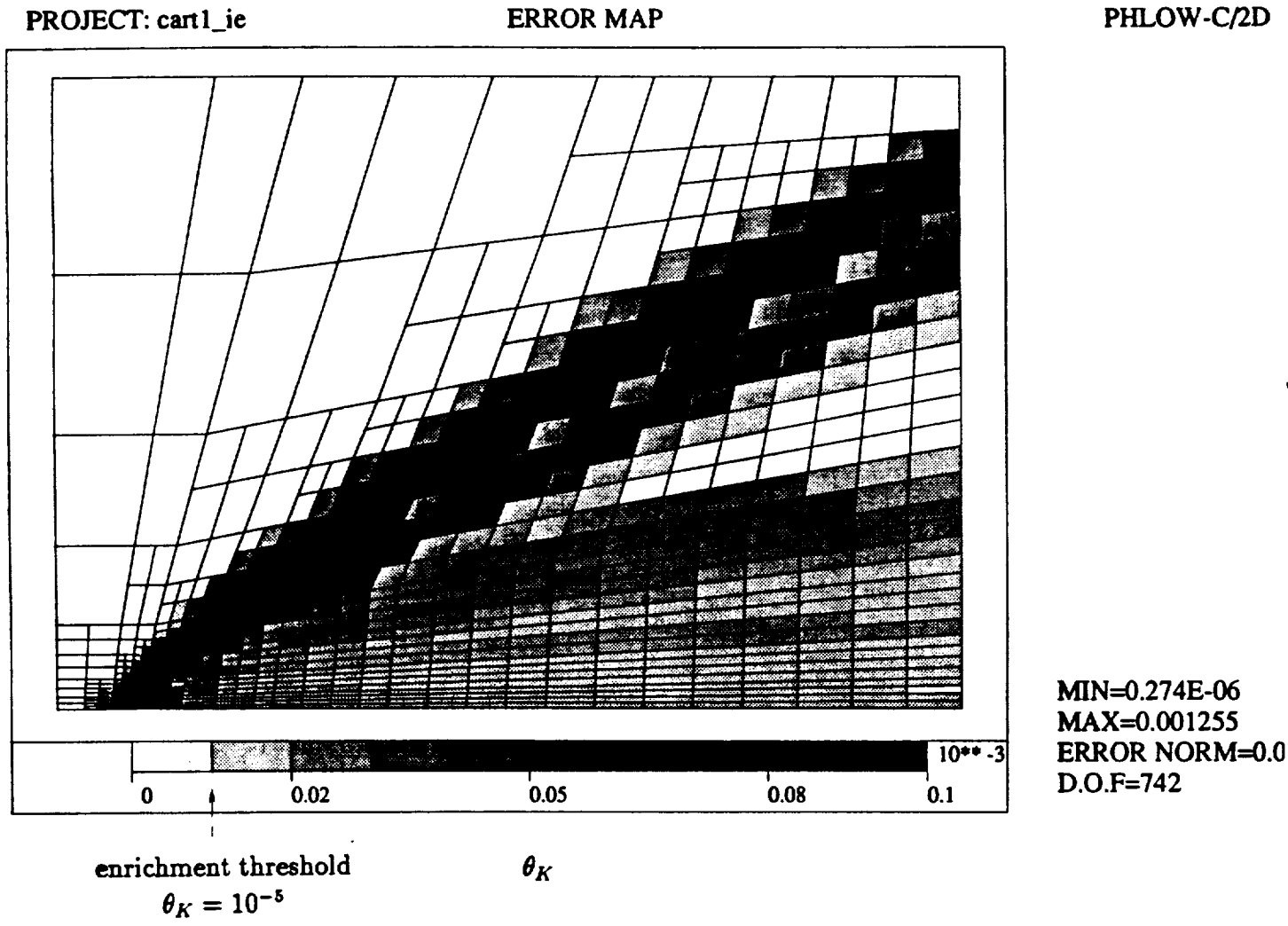


Figure 4.10: Map of the error indicators based on the element residual method.

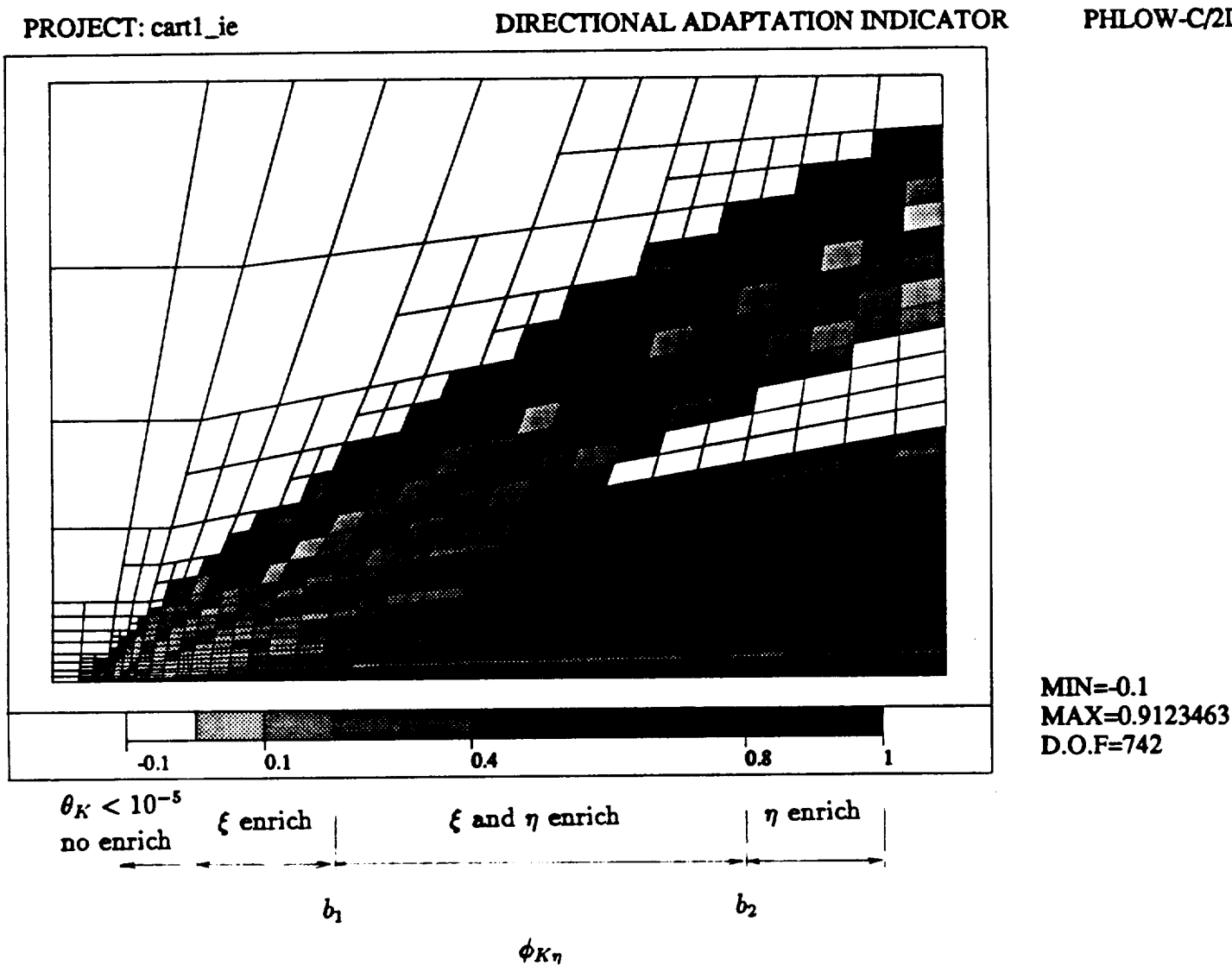


Figure 4.11: Map of the directional adaptation indicator based on the residual error function. (User specified values of b_1 and b_2 control the direction of p-enrichment)

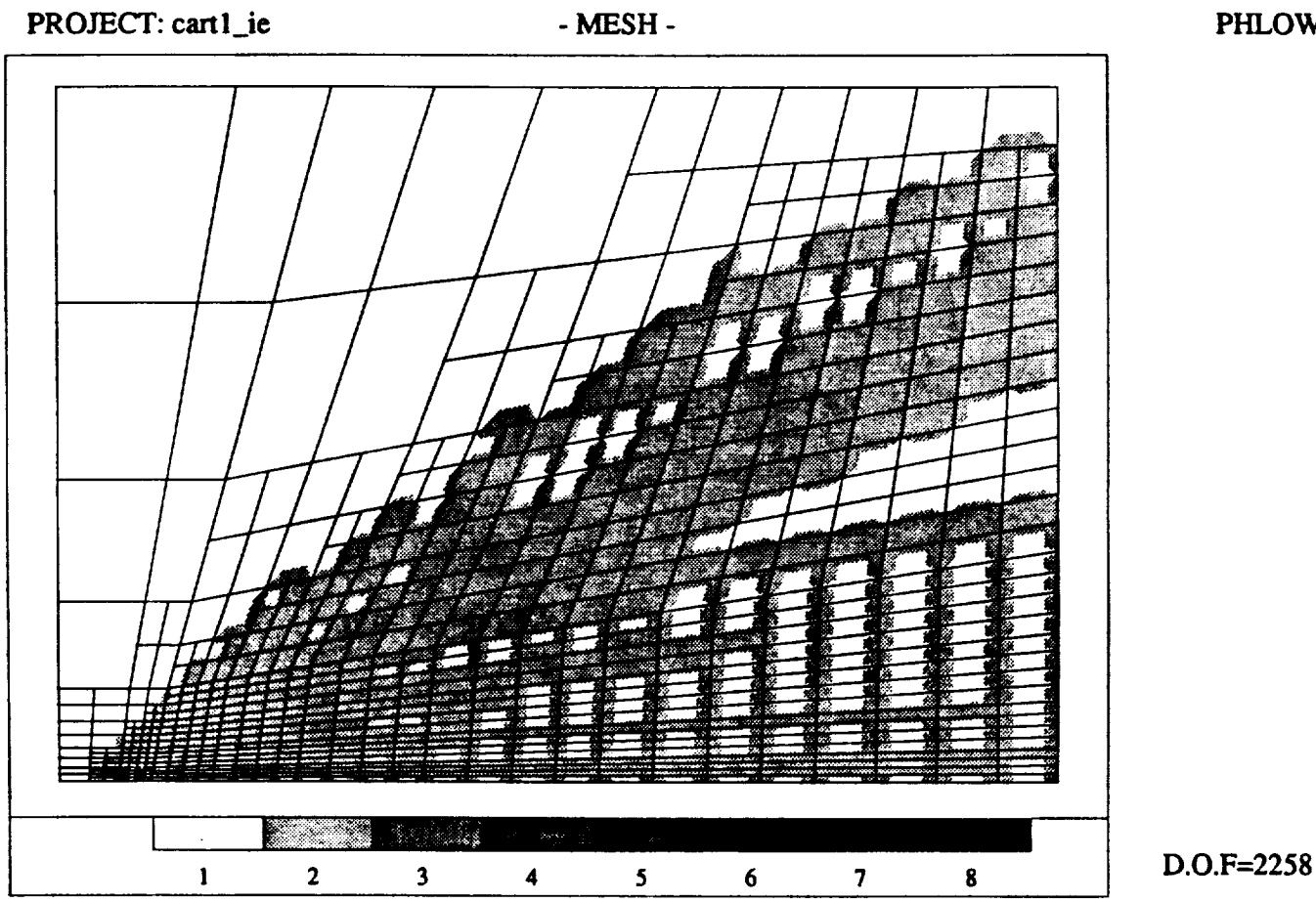


Figure 4.12: Enriched mesh according to residual-based $\phi_{K\eta}$: $b_1 = 0.2, b_2 = 0.8$

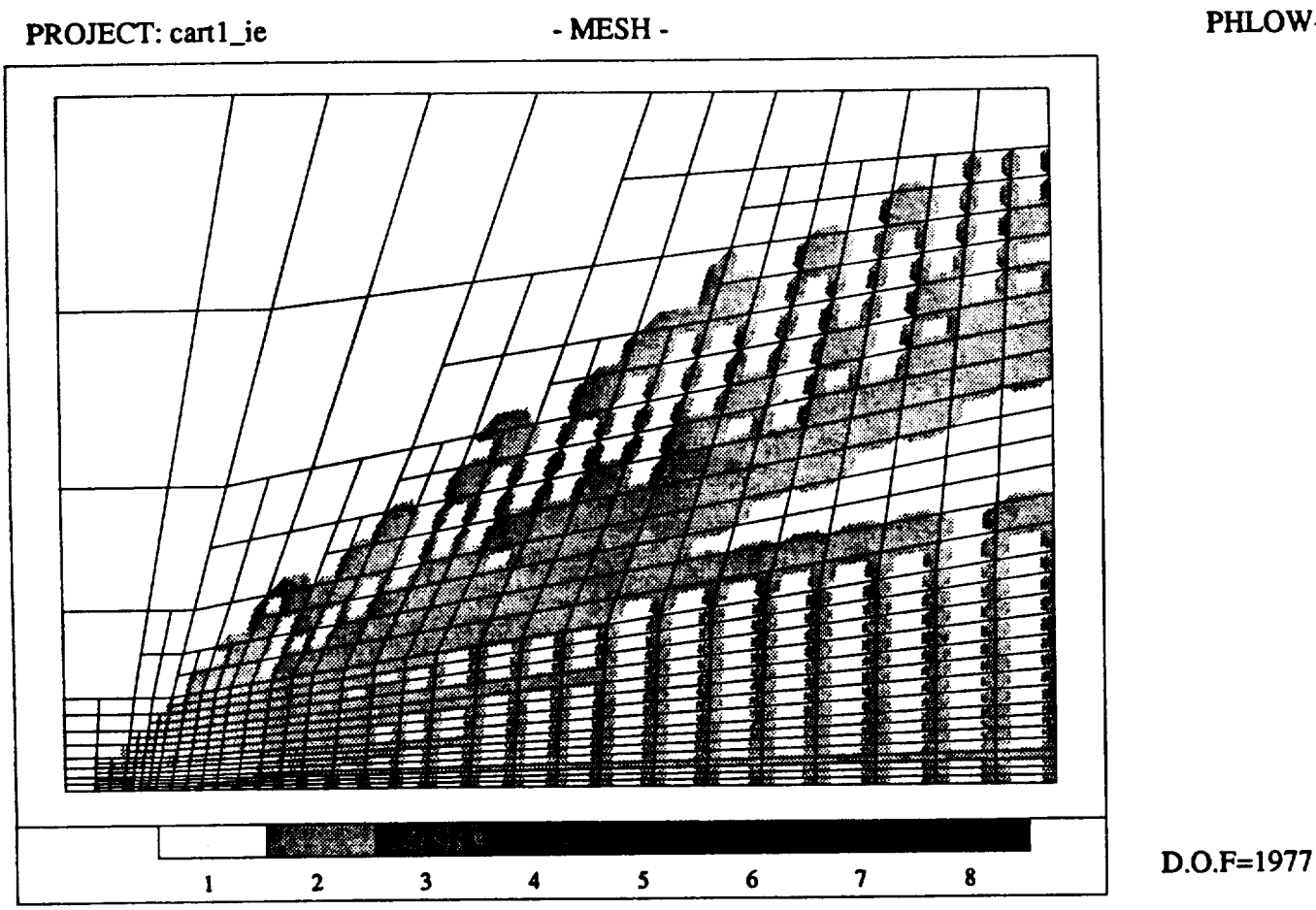


Figure 4.13: Enriched mesh according to residual-based $\phi_{K\eta}$: $b_1 = 0.3, b_2 = 0.7$

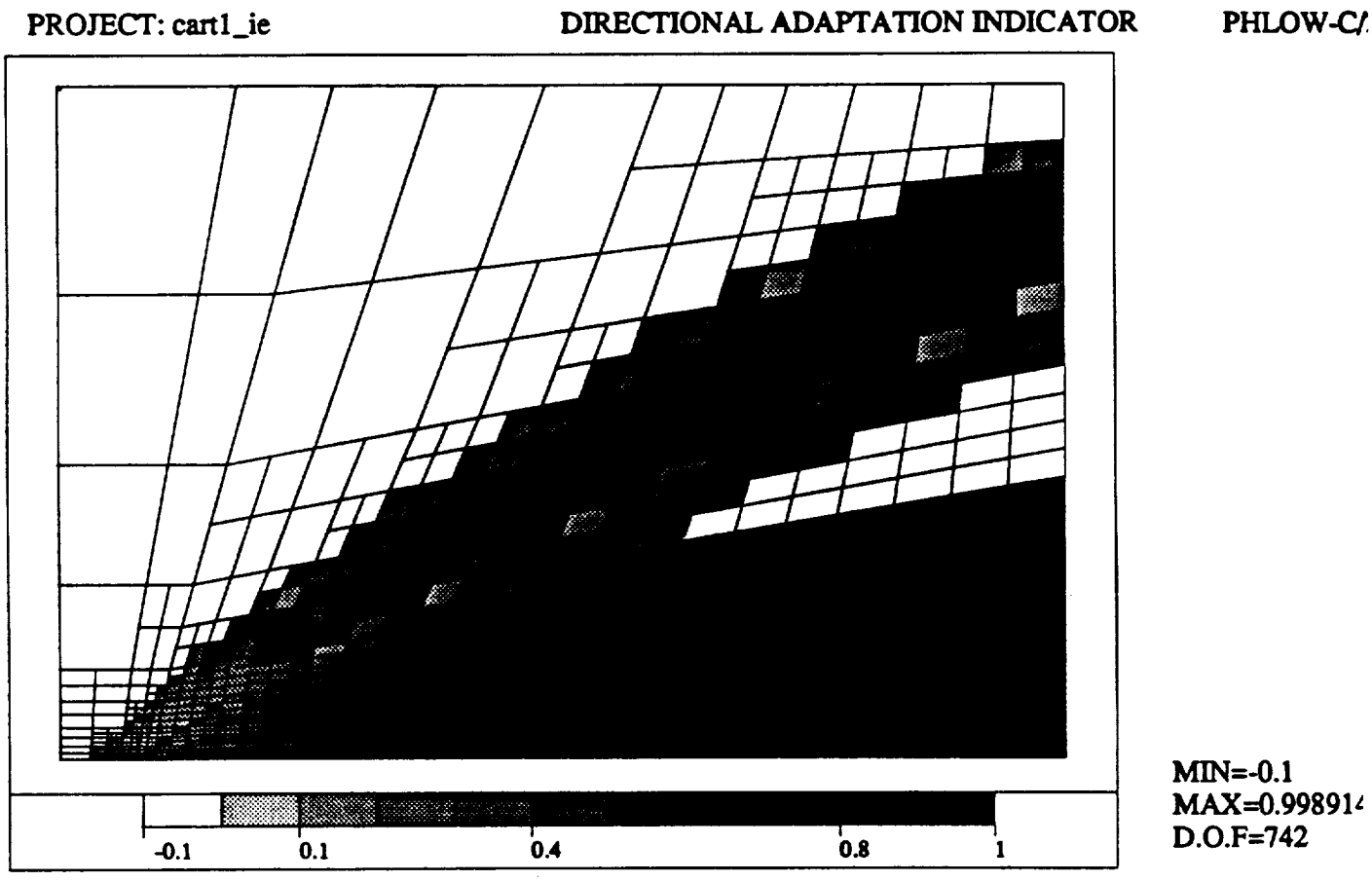


Figure 4.14: Map of the directional adaptation indicator based on the solution gradients.

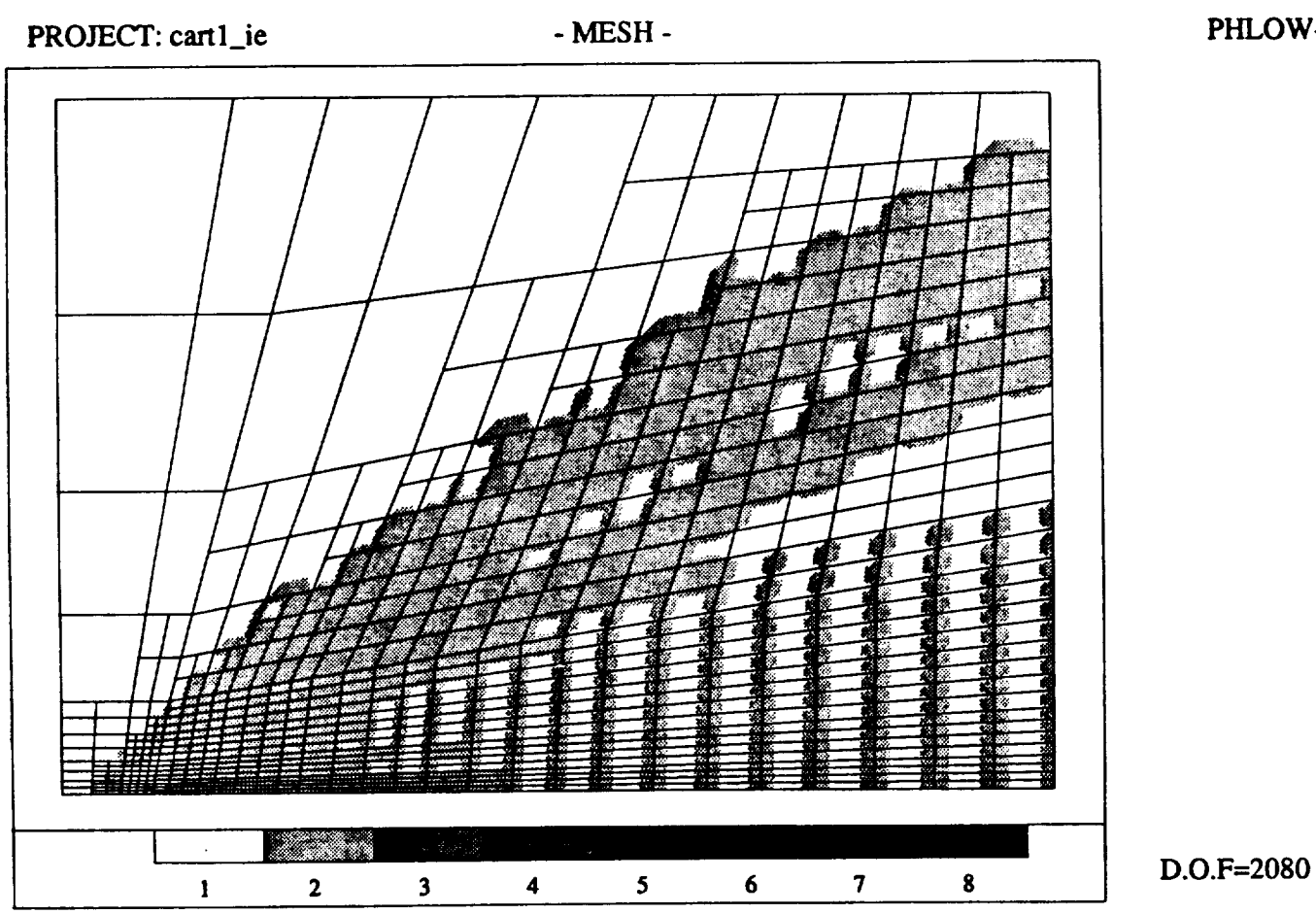


Figure 4.15: Enriched mesh according to $\phi_{K\eta}$ based on solution gradient: $b_1 = 0.2, b_2 = 0.8$

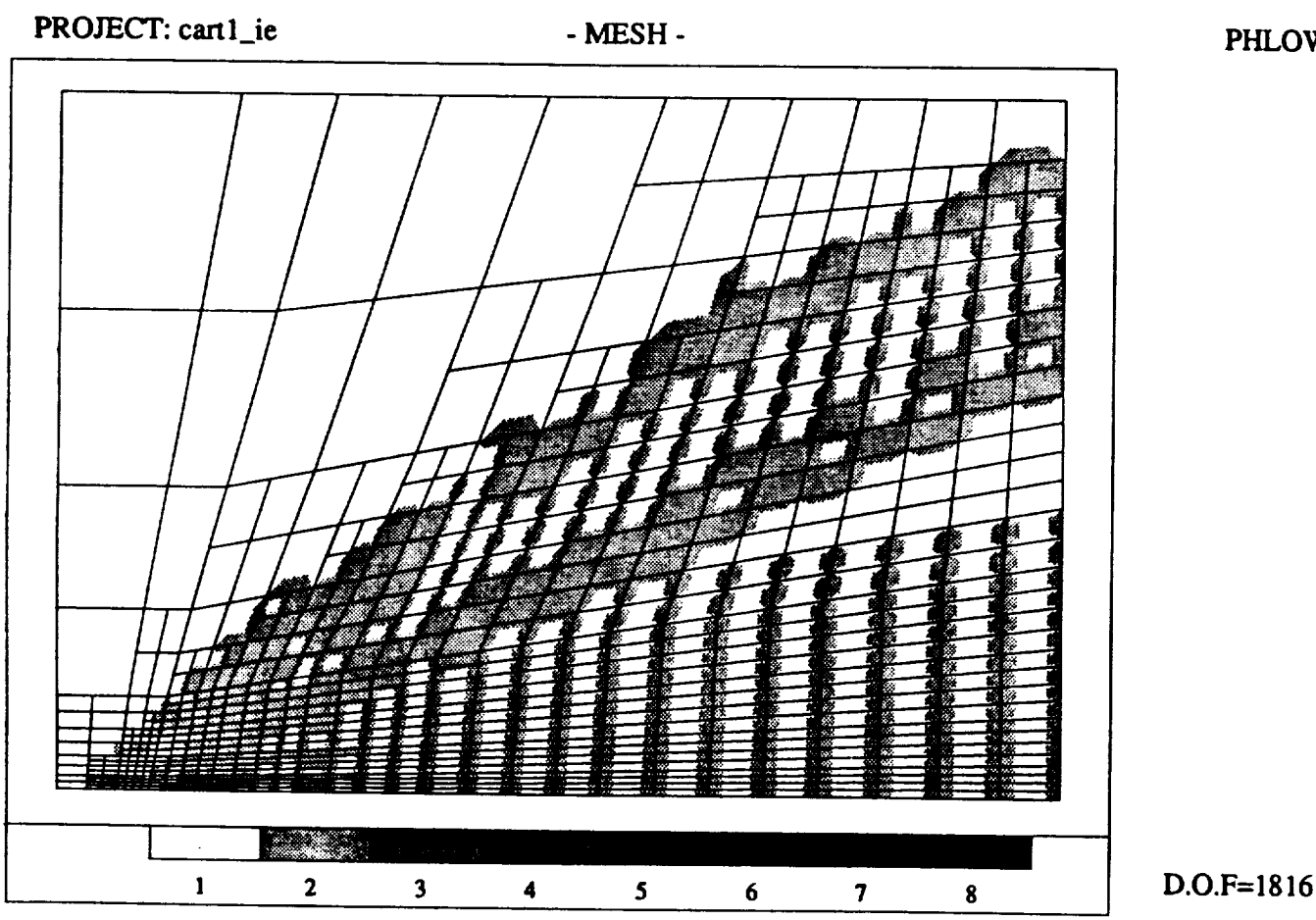


Figure 4.16: Enriched mesh according to $\phi_{K\eta}$ based on solution gradients: $b_1 = 0.3, b_2 = 0.7$

4.3 Adaptive Strategies

Once the distribution of errors is known, the decision must be made as to which elements should be refined. The h-p data structure allows two kinds of refinement: breaking elements (h-refinement) and enriching their spectral orders (p-refinement). As mentioned previously, the best improvement of the accuracy of finite element approximations can be achieved if a combination h- and p-refinements are performed. The problem of an optimal choice between h and p is, in general, still an open question. In the case of viscous flow problems, numerical experience suggests that p-enrichment is most desirable in boundary layer zones while in the rest of the computational domain the best choice is a linear approximation with h-refinements. The procedure of adapting meshes that we use in practice consists of performing exclusively h-refinements for steady state solutions until all the flow features have been resolved. Then, at the last stage of solving the problem, we enrich elements along the solid wall boundaries to provide high resolution of the viscous features of the flow, such as the skin friction and the heat flux.

As an adaptive strategy for h-refinements we adopted the well known strategy of equilibrating errors which leads to optimal meshes in the case of elliptic problems. The procedure of adapting a mesh based on this strategy consists of the following steps:

1. Integrate the solution forward in time until steady state is reached.
2. Determine the distribution of the error indicators e_i, $i = 1, \ldots, N$, N being the number of elements.
3. Refine the elements with the error larger than a certain percentage α_1 of the maximum error $e_{\max}$:

   $$e_i > \alpha_1 \cdot e_{\max}$$

   and unrefine the elements with errors less than $\alpha_2 \cdot e_{\max}$.
4. Go to 1.

The parameters α_1 and α_2 are user-specified and their values are assigned based on numerical experience. One practical selection of α_1 and α_2 is based on a percentage of the current number of elements to be unrefined and refined respectively. The general guideline is that in the case of flow problems, one should be rather generous with refinements and conservative with unrefinements. If smaller (refined) elements do not entirely cover shock regions, e.g. a shock across elements with different level of h-refinements, the small-large interelements may result in a misdisplacement of the shock.

5 Implicit/Explicit Procedures

During the first extension phase of this project an adaptive implicit/explicit method for the solution of viscous compressible flow has been implemented. This method is being used mainly to increase the efficiency of the *h-p* adaptive Navier-Stokes solver. The algorithm is based on the general family of Taylor-Galerkin models, with several parameters controlling the actual implicitness of the scheme, and is combined with *h-p* mesh adaptation and adaptive selection of implicit and explicit zones. Several criteria for the selection of these zones have also been studied. The theoretical formulation of the implicit/explicit method has been accomplished previously (see reference [31]), and some numerical results have been obtained using a different code with *h*-refinement capability. In the present work, the implementation of the implicit/explicit method is based on this experience, and the major task is to extend it for *hp*-adaptive procedure.

The basic idea of implicit/explicit algorithms is to combine the two methods to take advantage of the superior features of each. The major advantage of the explicit method is that element computations are relatively cheap and simple. Unfortunately this method suffers from stability limitations of the time step, which in some problems leads to prohibitively large numbers of time steps.

The implicit algorithm allows for an application of larger time steps than the explicit method. Moreover, due to the existence of implicit boundary terms, it offers easy and straightforward control of natural boundary conditions, particularly those involving the viscous fluxes. An additional advantage is that with larger time steps no explicit artificial dissipation is necessary, which is very important in the calculation of boundary fluxes, particularly wall heating rates. The major disadvantage of implicit methods is a much higher cost of element operations and a more complex and expensive solution of the resulting system of equations.

In this section, the formulation and numerical implementation of an adaptive implicit/explicit algorithm for compressible flows is presented. The algorithm is based on the general family of Taylor-Galerkin methods discussed in Section 2.

5.1 Formulation of implicit/explicit schemes

The algorithm of the implicit/explicit scheme must be designed so as to preserve stability, the conservative properties, and the required order of approximation. The procedure described below has been implemented in the code.

We begin by partitioning the domain Ω into subdomains $\Omega^{(E)}$ and $\Omega^{(I)}$ where explicit

and implicit schemes are to be applied, respectively, and

$$\Omega^{(E)} \cap \Omega^{(I)} = \Gamma_{EI} \ , \qquad \Omega^{(E)} \cup \Omega^{(I)} = \Omega$$

It is convenient to assume that the interface between the two regions coincides with the element boundaries.

It can easily be observed that the differential equations to be solved on the two subregions are different due to different implicitness parameters applied in each zone: $\alpha^{(I)}, \beta^{(I)}, \gamma^{(I)}$ in the implicit zone and $\alpha^{(E)}, \beta^{(E)}, \gamma^{(E)} = 0$ in the explicit zone . Therefore, the variational formulation (2.41), based on the assumption of constant implicitness parameters, cannot be applied to the domain Ω. Instead, it can be applied separately to each subdomain with additional continuation conditions across the interface. These conditions represent continuity of the solution and satisfaction of the conservation laws across the interface and are of the form:

$$\left.\begin{aligned} \boldsymbol{U}^{(E)} &= \boldsymbol{U}^{(I)} \\ \boldsymbol{F}_i^{(E)C} &= \boldsymbol{F}_i^{(I)C} \\ \boldsymbol{A}_i^{(E)} &= \boldsymbol{A}_i^{(I)} \\ \boldsymbol{F}_n^{(E)V} &= \boldsymbol{F}_n^{(I)V} \end{aligned}\right\} \text{ on } \Gamma_{EI} \tag{5.1}$$

where index n refers to the outward normal for the corresonding region ($\boldsymbol{n}^{(E)} = -\boldsymbol{n}^{(I)}$). The continuity requirement also pertains to the test function, so that $\boldsymbol{V}^{(E)} = \boldsymbol{V}^{(I)} = \boldsymbol{V}$. Note that for general weak solutions of Euler equations the solution $\boldsymbol{U}$ need not be continuous across the interface. However, for regularized problems and finite element interpolation, the continuity of $\boldsymbol{U}$ is actually satisfied.

If the variational statement is formulated for this problem, then in addition to interior integrals for each subdomain and regular boundary integrals, jump integrals across the interface appear in the formulation. These additional interface terms on the right-hand side are of the form:

$$\int_{\Gamma_{IE}} -\Delta t \left[\boldsymbol{F}_n^{(I)Cn} + \boldsymbol{F}_n^{(E)Cn}\right] \cdot \boldsymbol{V} + \Delta t \left[\boldsymbol{F}_n^{(I)Vn} + \boldsymbol{F}_n^{(E)Vn}\right] \cdot \boldsymbol{V}$$

$$+\frac{\Delta t^2}{2} \left[\left(1 - 2\alpha^{(I)}\right) \boldsymbol{A}_n^{(I)} \boldsymbol{F}_{,j}^{(I)Cn} + \left(1 - 2\alpha^{(E)}\right) \boldsymbol{A}_n^{(E)} \boldsymbol{F}_{,j}^{(E)Cn}\right] \cdot \boldsymbol{V} ds$$

On the left-hand side of the variational formulation additional interface terms are of the

form:

$$\int_{\Gamma_{IE}} \Delta t \left[\alpha^{(I)} \boldsymbol{A}_i^{(I)n} \Delta u^{(I)} n_i^{(I)} + \alpha^{(E)} \boldsymbol{A}_i^{(E)n} \Delta \boldsymbol{U}^{(E)} n_i^{(E)}\right] \cdot \boldsymbol{V}$$

$$- \Delta t \left[\gamma^{(I)} \left(\boldsymbol{R}_{ij}^{(I)} \Delta \boldsymbol{U}_j^{(I)} n_i^{(I)} + \boldsymbol{P}_i^{(I)} \cdot \Delta \boldsymbol{U}^{(I)} n_i^{(I)}\right) + \gamma^{(E)} \left(\boldsymbol{R}_{ij}^{(E)} \Delta u_j^{(E)} n_i^{(E)} + \boldsymbol{P}_j^{(E)} \Delta \boldsymbol{U}^{(E)} n_i^{(E)}\right)\right] \cdot \boldsymbol{V}$$

$$- \frac{\Delta t^2}{2} \left[\left(1 - 2\alpha^{(I)}\right) \beta^{(I)} \boldsymbol{A}_i^{(I)} \boldsymbol{A}_j^{(I)} \Delta \boldsymbol{U}_j^{(I)} n_i^{(I)} + \left(1 - 2\alpha^{(E)}\right) \beta^{(E)} \boldsymbol{A}_i^{(E)} \boldsymbol{A}_j^{(E)} \Delta \boldsymbol{U}_j^{(E)} n_i^{(E)}\right] \cdot \boldsymbol{V} ds$$

or, after reinterpretation of the linearized terms:

$$\int_{\Gamma_{IE}} \Delta t \left[\alpha^{(I)} \Delta \boldsymbol{F}_n^{(I)C} + \alpha^{(E)} \Delta \boldsymbol{F}_n^{(E)C}\right] \cdot \boldsymbol{V} - \Delta t \left[\gamma^{(I)} \Delta \boldsymbol{F}_n^{(I)V} - \gamma^{(E)} \Delta \boldsymbol{F}_n^{(E)V}\right] \cdot \boldsymbol{V}$$

$$- \frac{\Delta t^2}{2} \left[(1 - 2\alpha^{(I)}) \beta^{(I)} \boldsymbol{A}_n^{(I)} \Delta \boldsymbol{F}_{j,j}^{(I)C} + (1 - 2\alpha^{(E)}) \beta^{(E)} \boldsymbol{A}_n^{(E)} \Delta \boldsymbol{F}_{j,j}^{(E)C}\right] \cdot \boldsymbol{V} ds$$

In order to enforce interface conditions, the values of consecutive terms in the above formulas should be prescribed using equation (5.1). In this way, two first terms on the right hand side are set to zero. However, for other terms the direct application of interface conditions is impossible because of different coefficients for implicit and explicit components. *In the present implementation of this scheme we set all the interface components to zero.* This procedure preserves the continuity of fluxes and time accuracy across the interface up to the first order. Note that the artificial dissipation contributions, not presented here, are handled in exactly the same way as the natural viscosity.

5.2 Selection of implicit and explicit zones

The basic criterion for selection of implicit and explicit zones is simple: for a given time step all nodes which violate the stability criterion for an explicit scheme should be treated with the implicit scheme. According to this criterion, several options for an automatic adaptive selection of implicit/explicit zones were implemented:

1. User-prescribed time step Δt:

   Within this option, the user prescribes the time step. All nodes satisfying stability criterion for the explicit scheme (with s certain safety factor) are explicit. This means that all the elements connected to these nodes are treated with the explicit scheme. On all other elements the implicit scheme is applied.

2. Prescribed maximum CFL number:

   In this option, the user prescribes the maximum CFL number that can occur for elements in Ω. The time step is automatically selected as the maximum step satisfying

this condition. The choice of a maximum CFL number may be suggested by the time accuracy arguments or the quality of results.

3. A prescribed percentage of the domain is implicit.

   In this version, the user specifies the fraction of the domain which is to be treated implicitly. The elements with the strongest stability limitation (usually the smallest ones) are treated implicitly, the others are explicit. The time step is selected to guarantee stability of the explicit zone.

4. Minimization of the cost of computations.

   This option has not been implemented in the code. It belongs to the advanced theory of the cost minimization, and it is included here to explicate some basic concepts which are crucial to the successful application of the implcit/explicit procedure. In this option, the time step and the implicit/explicit subzones are selected to minimize the cost of advancing the solution in time (say one time unit). The algorithm is based on the fact that, for an increased time step, an increasing number of elements must be analyzed with the (expansive) implicit algorithm. The typical situation is presented in Fig. 5.1, which shows for different time steps the relative number of nodes that must be treated with the implicit scheme (to preserve stability). On the abscissa, the Δt_{FE} denotes the longest time step allowable for the fully explicit scheme (with certain safety factors). Δt_{FI} denotes the shortest time step requiring a fully implicit procedure. The relative number of implicit nodes increases as a step function from zero for $\Delta t \leq \Delta t_{FE}$ to one for $\Delta t \geq t_{FI}$. Now assume that the ratio r of the computational cost of processing one implicit node to the cost of processing one explicit node is given. This ratio can be estimated relatively well by comparing the calculation time of element matrices and adding, for implicit nodes, a correction for the solution of the system of equations. Then the reduction of the cost of advancing the solution in time with the implicit/explicit scheme, as compared to the fully explicit scheme, is given by the formula:

$$R(\Delta t) = \frac{\Delta t_{FE}}{\Delta t}\left(n^{(E)} + r n^{(I)}\right)$$

   Typical plots of the function $R(\Delta t)$ are presented in Fig. 5.2. Shown here are the two cases:

   (a) the case of a small difference between fully explicit and fully implicit time steps—an almost uniform mesh

   (b) the case of a large difference between fully explicit and fully implicit time steps

Note that in either case, restrictions on the length of the time step should be applied, for example, from the maximum CFL condition. Otherwise the cheapest procedure would always be to reach the final time with one implicit step.

From the plots in Fig. 5.2, the following observation can be made: for an essentially uniform mesh, the mixed implicit/explicit procedure does not provide savings of the computational cost—either a fully implicit or fully explicit scheme is the cheapest depending on the time step restriction. On the other hand, for very diverse mesh sizes the mixed procedure provides considerable savings. This means that the effectiveness of the mixed implicit/explicit scheme will be the best for large-scale computations with both very large and very small elements present in the domain. Some introductory numerical results confirming this observation are presented in Section 5.4. In the practical implementation of this method, the approximation of the function $R(\Delta t)$ is automatically estimated for a given mesh. Then, the time step corresponding to the smallest $R(\Delta t)$ is selected automatically (subject to additional constraints, in particular the $\mathrm{CFL}_{\max}$ constraint).

In our *hp* mesh adaptation procedure, all high order elements ($p \geq 2$) are handled by implicit scheme, i.e., all high order nodes are treated as implicit nodes regardless of the stability limitation. This is because, with hierarchial shape functions, and $p \leq 2$, even explicit procedure generates out-of-diagonal entries in the mass matrix. Since presently there are no efficient methods of lumping such a mass matrix, the explicit algorithm does not really improve efficiency for higher p orders. Therefore only the nodes of the linear elements are treated as candidates for implicit/explicit selection procedure. This implementation has the following two advantages: (1) since the high order elements are often used within boundary layers, application of implicit scheme in these regions provides faster convergence of the boundary fluxes and offers direct control of the natural boundary conditions; and (2) with the exclusion of high order elements from the explicit zone, the mass lumping procedure can be easily implemented and offers a very low computation cost in this region.

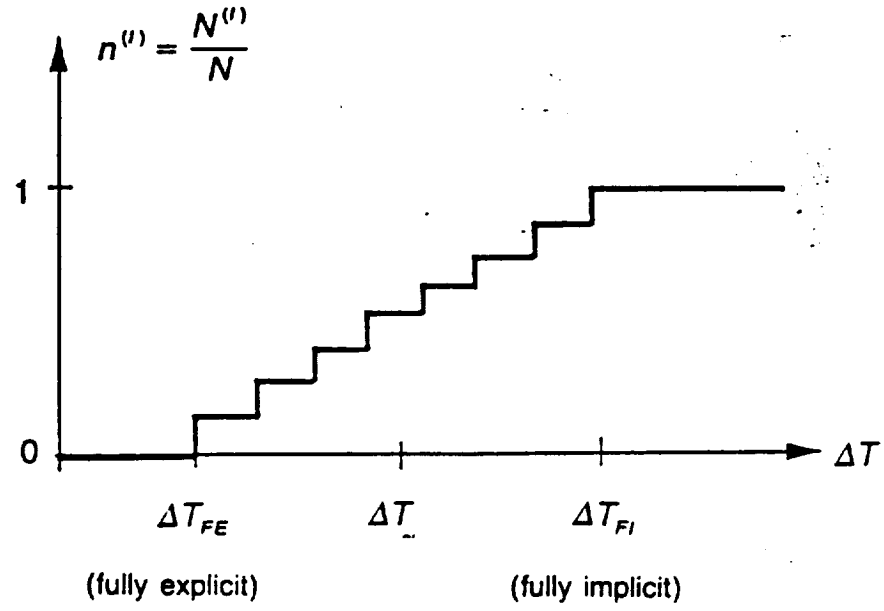


Figure 5.1: Relative number of implicit elements for increasing time step.

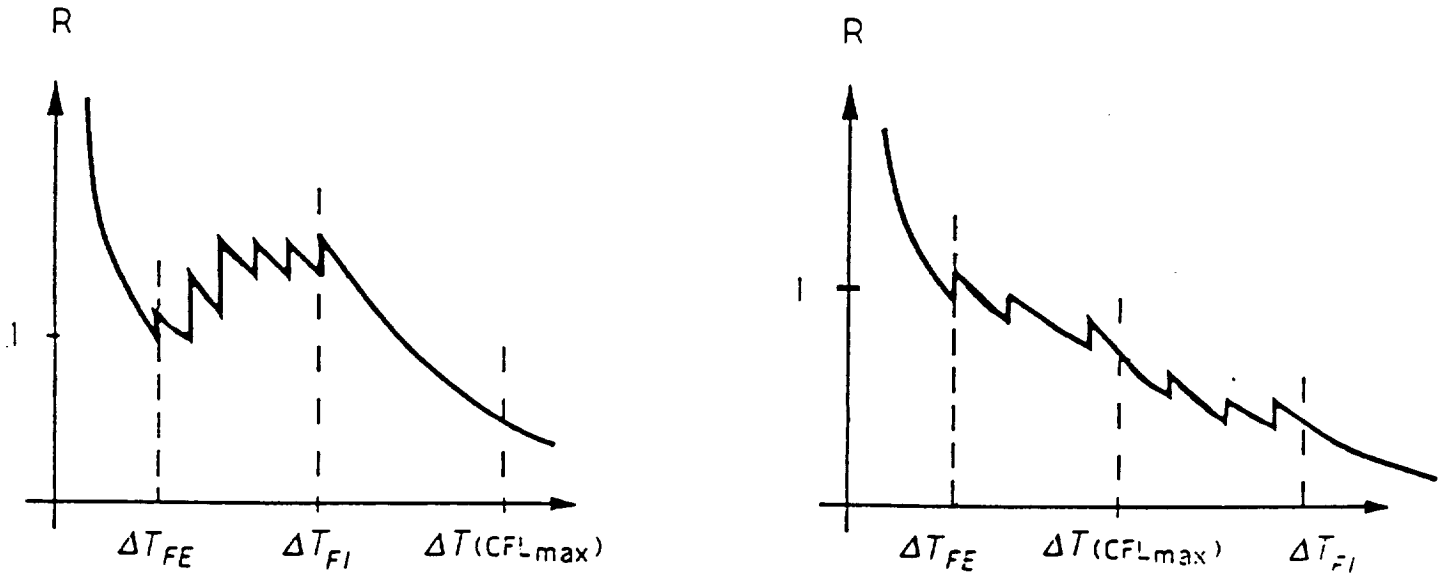


Figure 5.2: Reduction of the cost of computations due to implicit-explicit procedure.

The criteria described above are based purely on a stability analysis. The calculation of the stable explicit time step for each linear node is based on the most conservative estimation: the minimum value is taken from all the elements connected to this node and also the elements whose constrained nodes are computed from this node. Nodes are defined as implicit or explicit according to whether the explicit stability criterion is violated or not. Once the implicit and explicit nodes have been defined, implicit and explicit elements are selected. All linear elements which contain at least one explicit node are defined as explicit, the remaining elements become implicit.

5.3 Computational procedure

In our compressible flow solver, the implicit/explicit procedure is combined with an h-p adaptation scheme. The range of implicit/explicit zones is redefined after every mesh adaptation and between adaptation after every prescribed number of steps. At every time step, the implicit/explicit procedure results in a system of equations

$$(\boldsymbol{M} + \boldsymbol{K}^{(I)})\boldsymbol{U}^{n+1} = \boldsymbol{R}$$

where the stiffness matrix $\boldsymbol{K}^{(I)}$ has non-zero entries only for degrees of freedom in the implicit zone or for nodes with penalty-enforced boundary conditions. The solution procedure for the above system depends upon whether the consistent or lumped matrix is used, and the capability of the solver used for the system of equations, for example, whether it can solve efficiently a system with block diagonal structure.

Based on the user specified implicit/explicit zone selection criterion described in previous section, the current version of the implicit/explicit solver uses a two-pass procedure: the first pass is a loop through explicit elements to solve for the solution of explicit nodes by the lumped mass method, and the second pass solves the remaining implicit degrees of freedom with a direct frontal solver. Incorporating this algorithm within the h-p data structure, the procedure can be listed as follows:

FIRST PASS

1 Loop through the explicit elements:

- **1.1** form the unconstrained consistent mass matrix and right-hand-side (RHS) vector
- **1.2** modify the mass matrix and RHS vector for contrained nodes
- **1.3** lump the mass matrix and assemble RHS vector for explicit nodes

2 Obtain the solution for explicit nodes

SECOND PASS

3 Loop through all elements:

3.1 For explicit elements containing implicit nodes:

3.1.1 form the unconstrained mass matrix and RHS vector

3.1.2 modify the unconstrained mass matrix and RHS vector for constrained nodes

3.1.3 lump the mass matrix

3.1.4 apply boundary conditions

3.1.5 for implicit nodes, store the stiffness matrix and RHS vector in a frontal solver format

3.2 For implicit elements:
Following the standard procedure, form the stiffness matrix and RHS vector, and then store them in frontal solver format

4 Obtain the solution for the implicit degrees of freedom using direct solver.

Remarks

1. The calculation of mass matrix in step (1.1) needs to be done only for a single solution component.

2. If the global lumped mass matrix is saved, it needs to be recomputed only after the mesh is adapted or the implicit/explicit zones are redefined.

3. The lumped mass matrix is obtained by summing the rows of the "constrained" consistent mass matrix. For boundary nodes in explixit elements, the lumping is performed before the application of the boundary conditions.

4. If the application of penalty-enforced boundary conditions produce off-diagonal entries to the stiffness matrix (e.g. solid-wall and no-flow boundary conditions), the boundary nodes are treated as implicit nodes and their solutions are solved in the second pass.

Numerical Illustration of Implicit/Explicit Procedures

The first example is the Carter flat plate problem solved on a mesh with 3 levels of h-adaptation. The density contours are shown in Fig. 5.3. Figs. 5.4a and 5.4b represent the meshes when 40 % and 80 % of the degrees of freedom are selected as implicit, respectively. The implicit elements are shaded dark, the explicit elements containing at least one implicit node or node on the boundary which gives off-diagonal entries are shaded light, and the

explicit elements containing only explicit nodes are not shaded. A complete set of timing test runs has also been performed on this problem. These test used the third implicit/explicit zone selection option—the user specifies percentage of the domain to be treated implicitly. The results are tabulated in Table 5.1, showing the percentage of implicit zone, the timestep size, Δt, the equivalent CFL number (based on fully explicit sheme with safety factor set to 1). and the cost reduction factor (also based on fully explicit scheme). When the implicit/explicit zone is selected based on the cost minimization with the limitation on the time step controlled by the condition CFL < 10, about 65 % of the nodes are treated as implicit nodes. The computation cost of reaching the steady-state solution is reduced by a factor of 2 with respect to the fully explicit algorithm.

The second example is a blunt body with incident shock (the detailed description of the problem is presented in Section 7). A relative coarse initial mesh consisting of 18 by 40 linear elements is used for the *inviscid* solution. The corresponding pressure contours are shown in Fig. 5.5. The stable time steps of each node for the explicit scheme (based on CLF $= 1$) are contoured in Fig. 5.6 and range from 0.0326 to 0.3408. The implicit and explicit zones selected for a given time step of 0.1 are plotted in Fig. 5.7. The corresponding cost reduction factor is about 0.71. The same mesh with 2-levels of h-adaptation for the *viscous* solution and the pressure contour are shown in Figs. 5.8 and 5.9, respectively. The stable time steps of the nodes for explicit scheme are in the range between 0.006727 and 0.3412. The implicit and explicit zones selected, shown in Fig. 5.10, are based on the second option with a specified maximum CFL of 3. For visual clarity, the drawing of element grid lines is suppressed and the colormap is adjusted so that the light, medium and dark shades represent the fully explicit elements, explicit elements containing implicit nodes, and fully implicit elements, respectively. The nodes treated implicitly cover about 37 percent of the domain, and cost reduction factor with respect to the fully explicit method is about 0.92. Compared to a fully implicit method (which was previously the only method available in this project) the cost reduction factor of the implicit/explicit method is about 0.14. It should be noted that, although the pressure contours show reasonable resolution of shocks, the clustering of the elements near the blunt body is still too coarse to resolve the flow features in the viscous layer. The numerical results and discussion using a finer mesh are presented in Section 7.

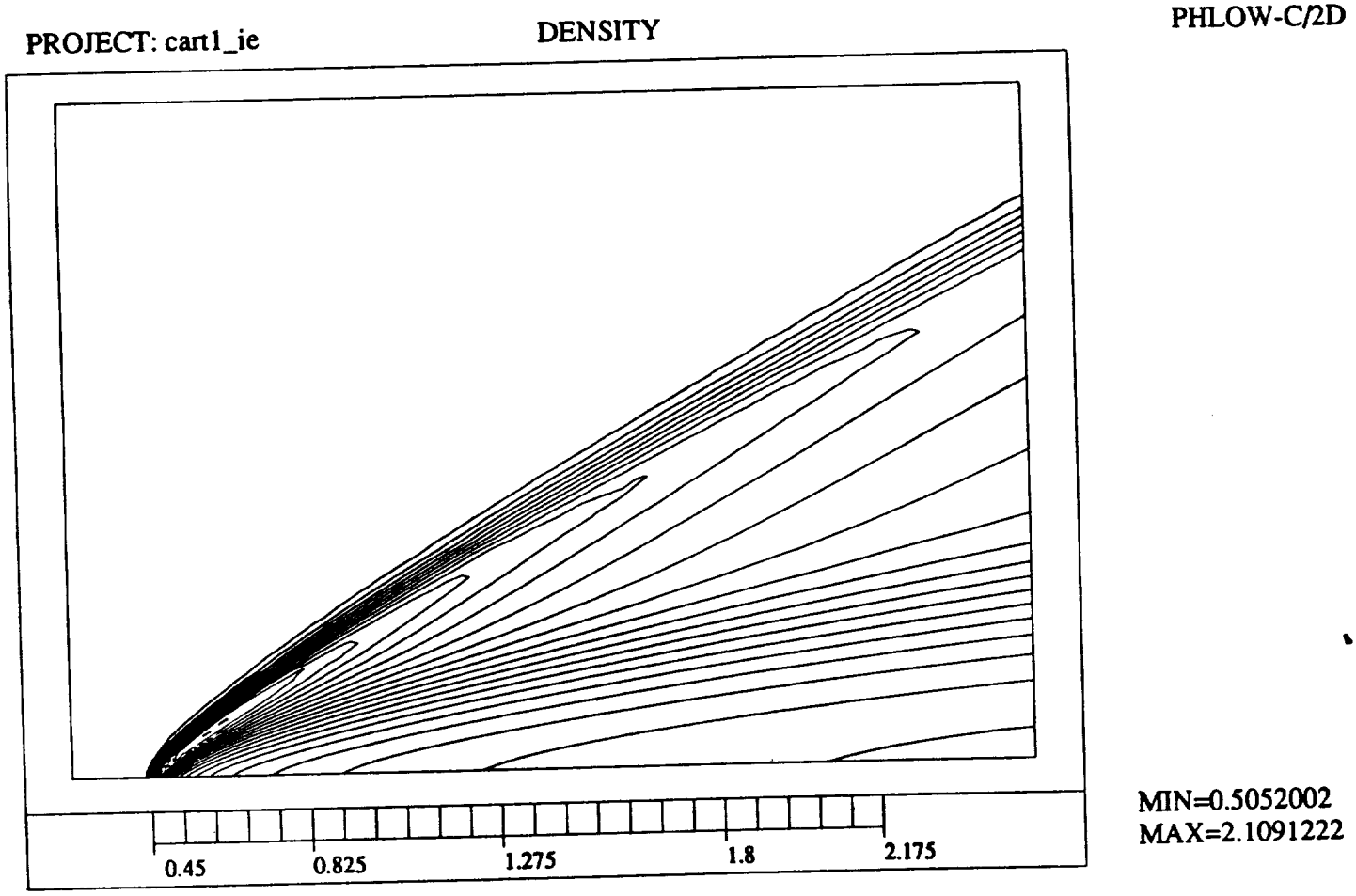


Figure 5.3: Density contours for Carter's problem on a mesh with third level of h-refinement.

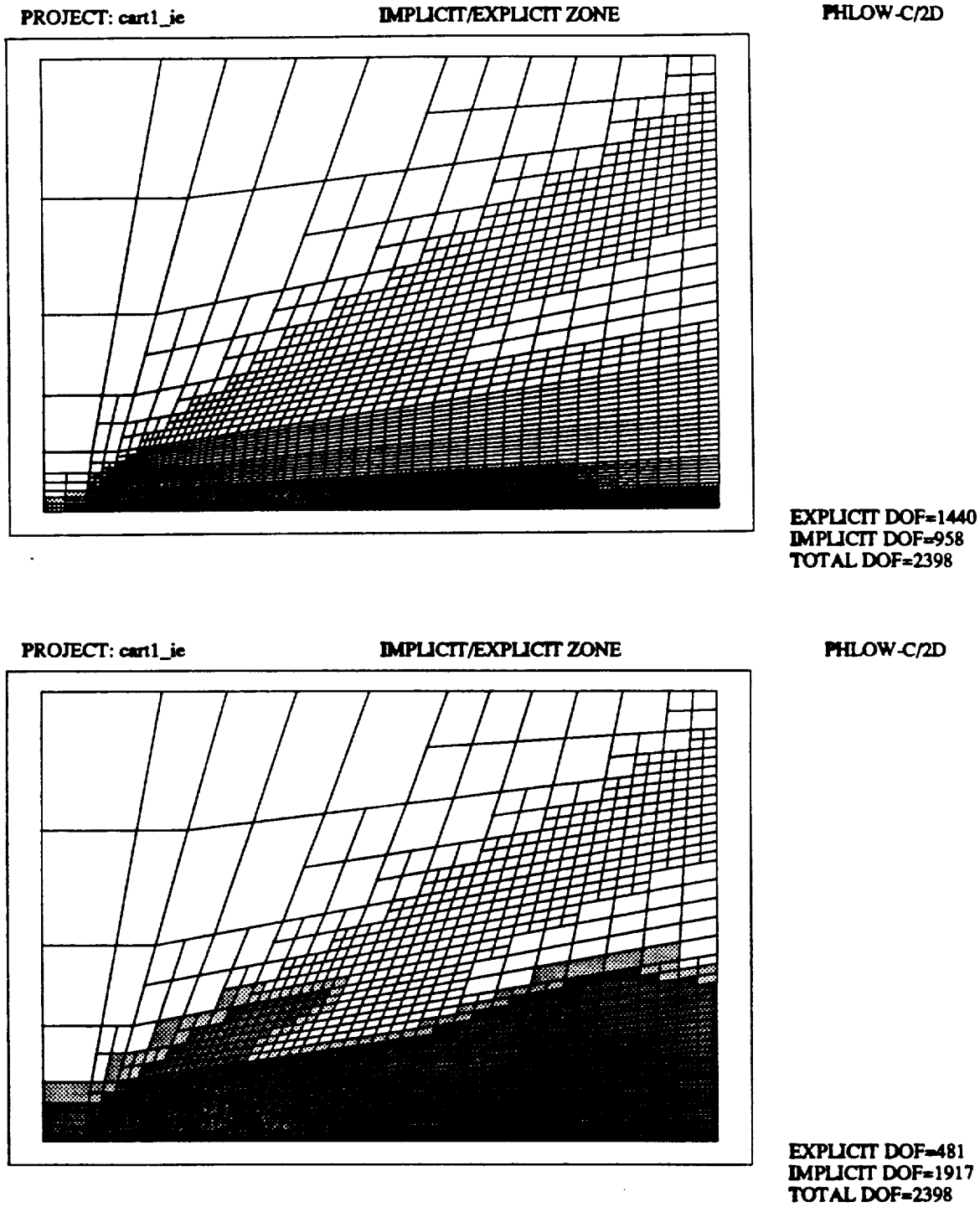


Figure 5.4: Implicit and explicit zones for the Carter flat plate problem (a) with 40 % of nodes treated implicitly and CFL $\cong$ 4.5 and (b) with 80 % of nodes treated implicitly and CFL $\cong$ 29.

% of Implicit	Δt	Equivalent CFL number	Cost Reduction
0	.000335 (Δt_{FE})	1	1
10	.000446	1.33	0.97
20	.000755	2.25	0.73
30	.000906	2.87	0.81
40	.001511	4.51	0.63
50	.002012	6.01	0.62
60	.002927	8.74	0.55
70	.004415	13.18	0.45
80	.009791	29.23	0.25
90	.011064	33.02	0.27
100	.1124 (Δt_{FI})	(335.5)	

Table 5.1

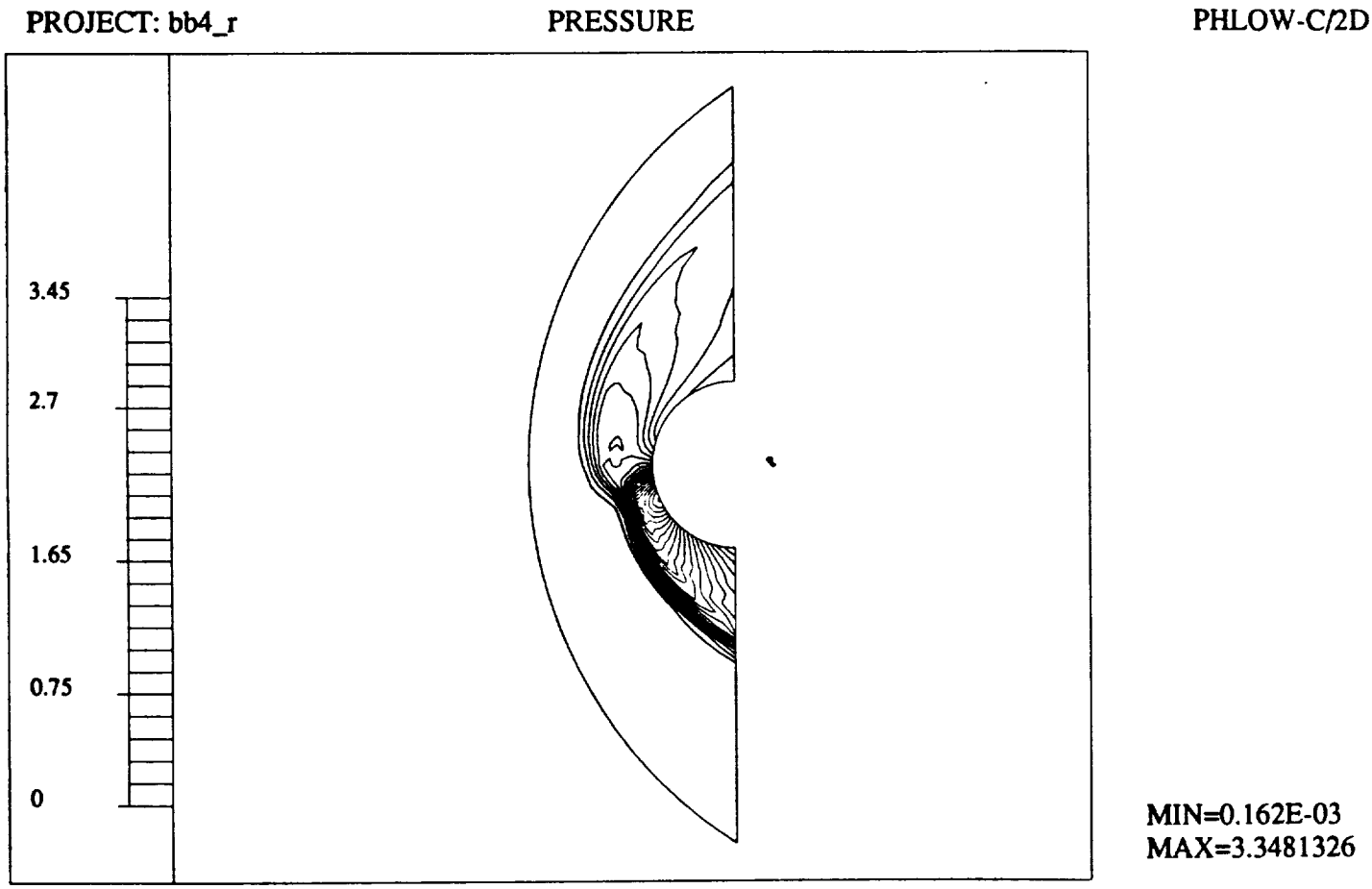


Figure 5.5: Pressure contours of inviscid solution on a coarse mesh.

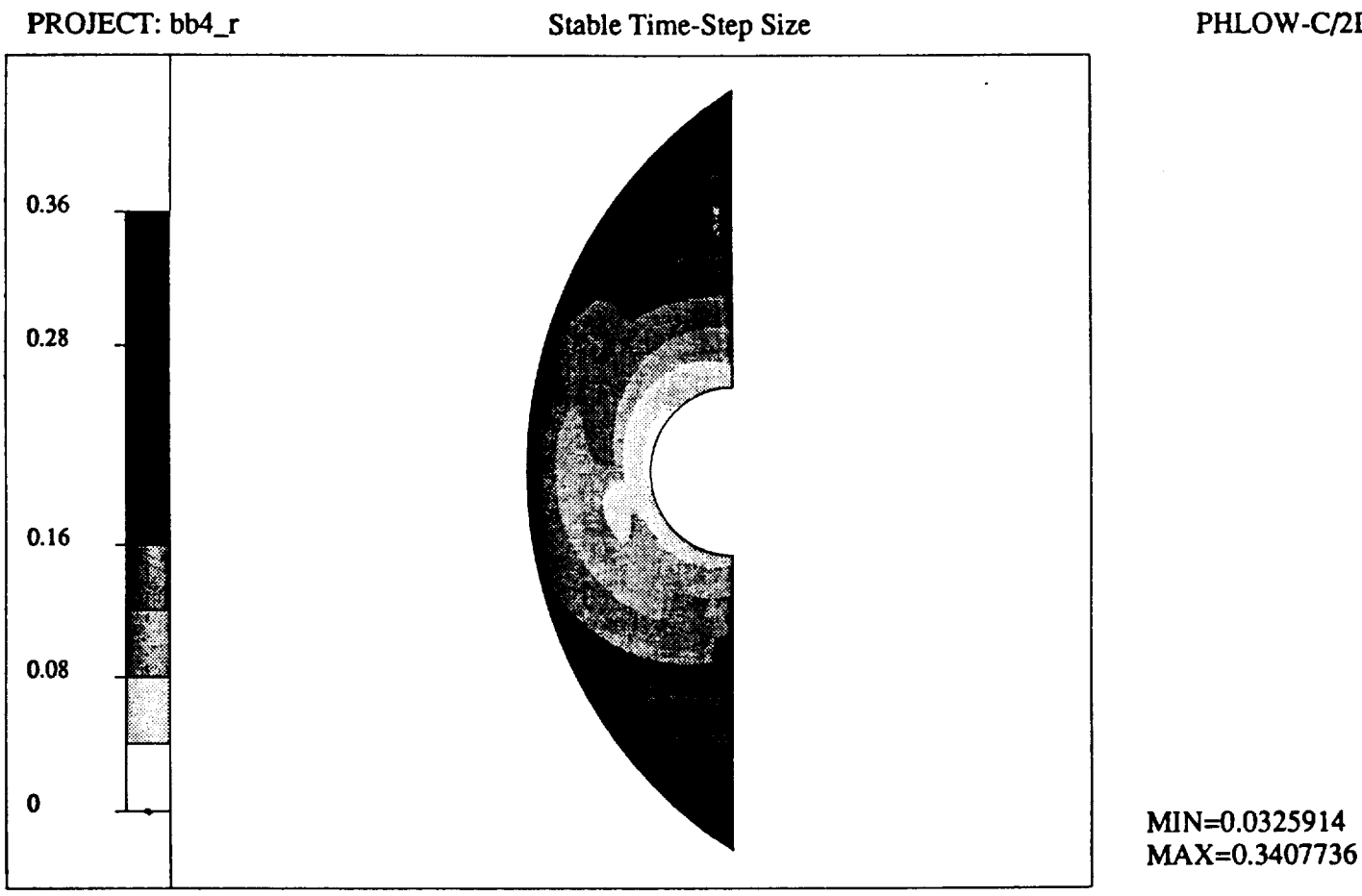


Figure 5.6: Contours of stable time-step sizes of nodes.

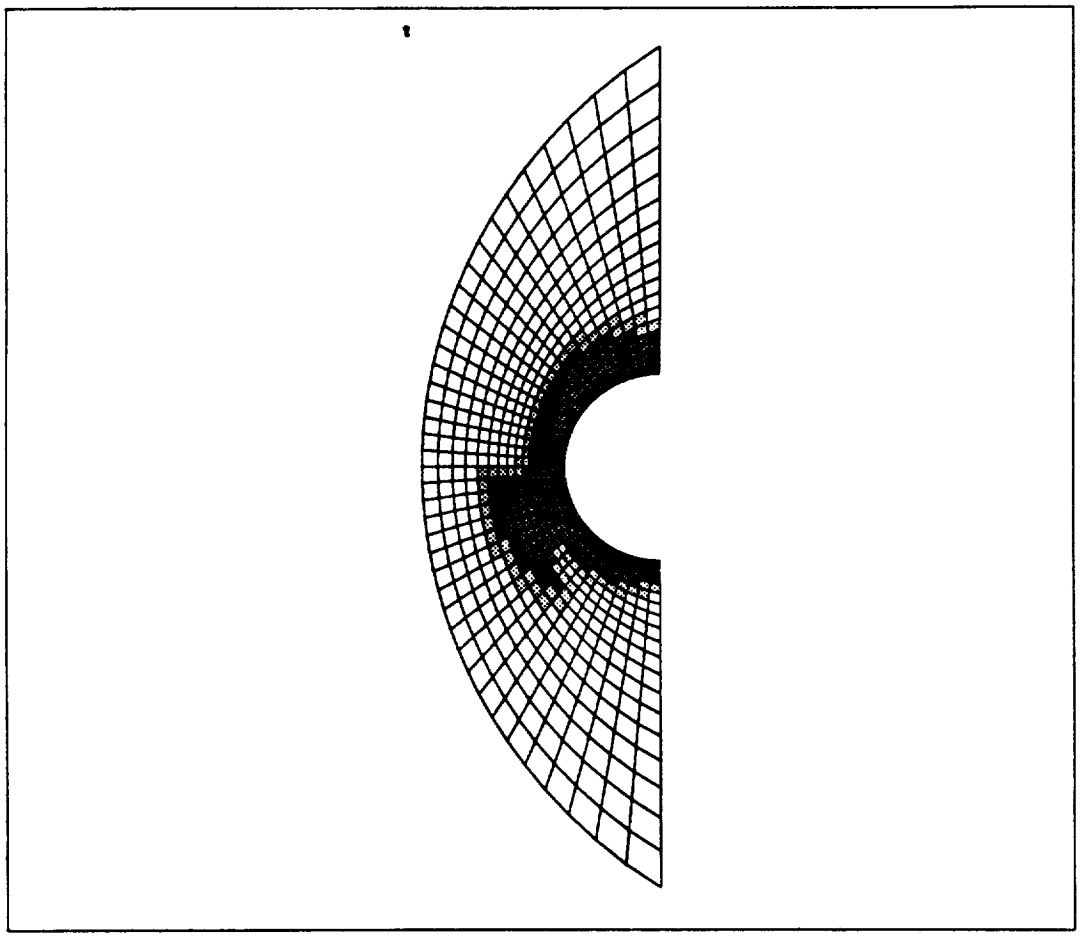


Figure 5.7: Implicit and explicit zones selected by specifying $\Delta t = 0.1$

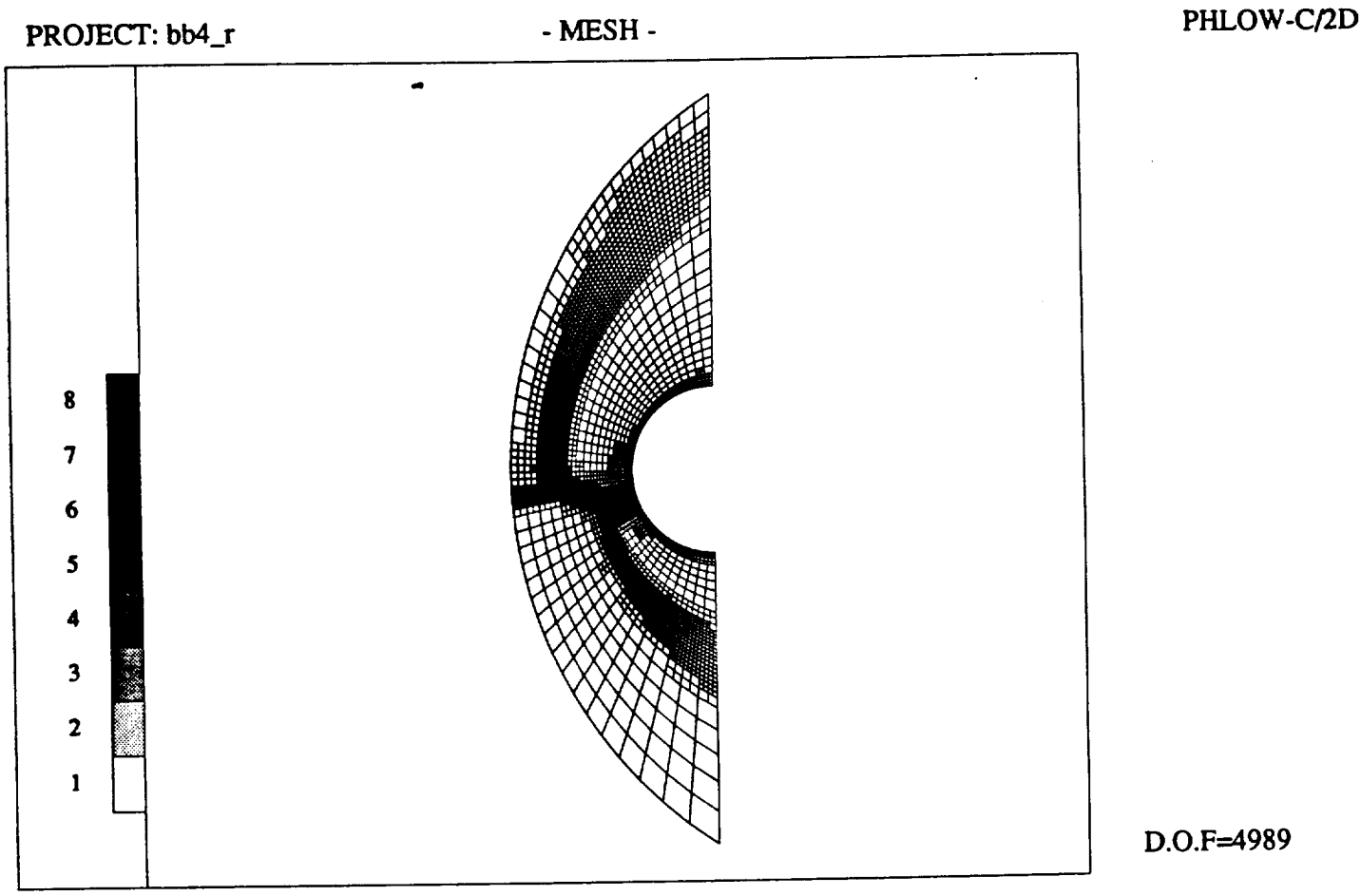


Figure 5.8: Two level h-adapted mesh for viscous solution.

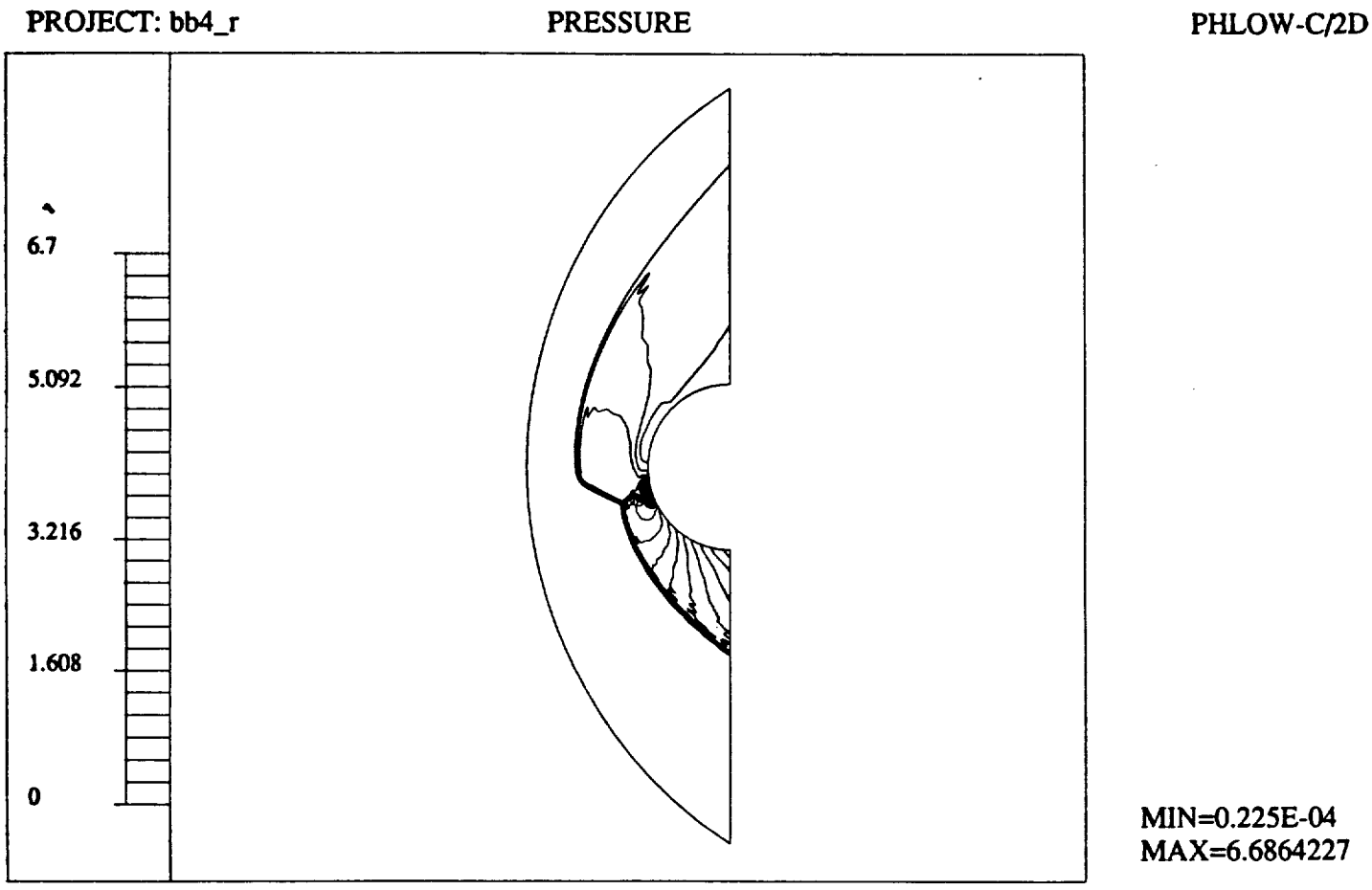


Figure 5.9: Pressure contours of viscous solution.

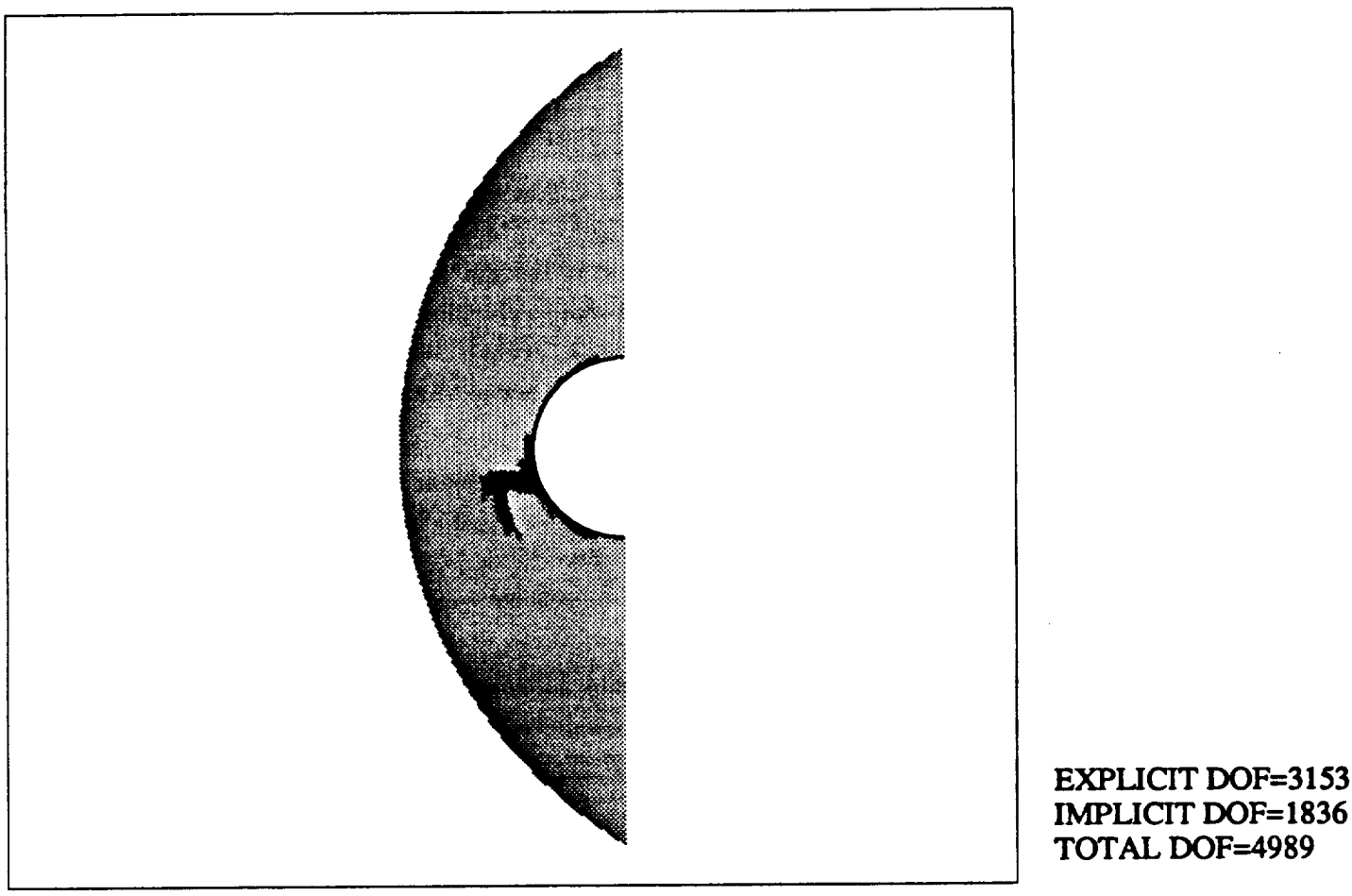


Figure 5.10: Implicit and explicit zones selected by specifying max $CFL = 3$.

6 Some Nonstandard Algorithms

In this section we present several nonstandard finite element procedures which are necessary to efficiently implement the *h-p* finite element methodology.

6.1 Integration and Underintegration Procedures

Solving the boundary value problems associated with the Taylor-Galerkin approximation and the approximation of the viscous step involves enforcing essential and mixed boundary conditions, such as the no-penetration boundary condition on solid walls. These boundary conditions are currently being enforced by means of a penalty function. It is well known that this approach requires special integration procedures guaranteeing that the number of integration points over any patch of elements on the boundary matches the number of degrees of freedom in this region. Otherwise, if the number of integration points exceeds the number of degrees of freedom, a locking effect may result when enforcing more boundary constraints than needed. Practically, this means that integration points must be associated with nodal locations and their number must be equal to the number of shape functions corresponding to a given node.

In two dimensions the usual way of dealing with this problem is to introduce appropriate underintegration on the boundary with mixed conditions, i.e., integration which for a p-th order one–dimensional boundary of the element introduces $p+1$ points with two of them at the endpoints. The Gauss-Lobbato integration scheme satisfies such conditions and is frequently used.

In three–dimensional *h-p* finite element methods, the situation is more complicated, since one may now encounter constrained nodes and nonuniform distributions of degrees of approximation p on the two–dimensional surface of the computational domian. Consider, for example, the situation presented in Fig. 6.1, where both active and constrained nodes are shown. If we take the element K_1, unconstrained and with uniform orders p, then a tensor product of $p+1$ one–dimensional Gauss-Lobatto integrations will have the proper number and distribution of integration points. If we consider, however, an element with nonuniform p's, like K_2, then no tensor product of one–dimensional procedures will be suitable as such products distribute integration points according to rectangular patterns not adequate for nonuniform distributions of degrees of freedom. The situation is more complicated for constrained elements, like K_3: integration points along the side adjacent to element K_4 must be chosen such that they coincide with integration points of the larger unconstrained neighbor. Otherwise, more conditions on the global shape functions would be imposed over both elements than the number of degrees of freedom. Figure 6.2 presents a possible distribution of integration points which would be satisfactory from the point of view of avoiding locking

effects.

The problem that emerges is to construct integration procedures which are accurate enough but which also satisfy the indicated restrictions on the numbers and locations of sampling points. To avoid this problem, note that the integration of penalty terms does not require any accuracy of the integration procedure: the penalty approach is an implicit way of imposing pointwise conditions on the solution (collocation). The only reason for using some specific integration procedures (i.e., with some accuracy) is that not only are penalty terms integrated on the boundary, but also other quantities (like stresses, etc.) are computed on the boundary which we wish to integrate with sufficient accuracy as well. This suggests that we simply use separate procedures for integrating penalty terms and all the remaining quantities. In the first case, use sampling points (not necessarily integration points) whose number and location corresponds to locations of nodes and their orders p, with (say) unit weights. In the second case, use ordinary numerical integration procedures, for instance Gaussian quadratures. An example of a choice of sampling points for integrating penalty terms is shown in Fig. 6.3: they are distributed uniformly along unconstrained sides and the interior of the element wall with their number being $p+1$ or $(p+1)^2$, respectively. For constrained sides, sampling points are chosen to coincide with those of the larger neighbor for which the common side is not constrained.

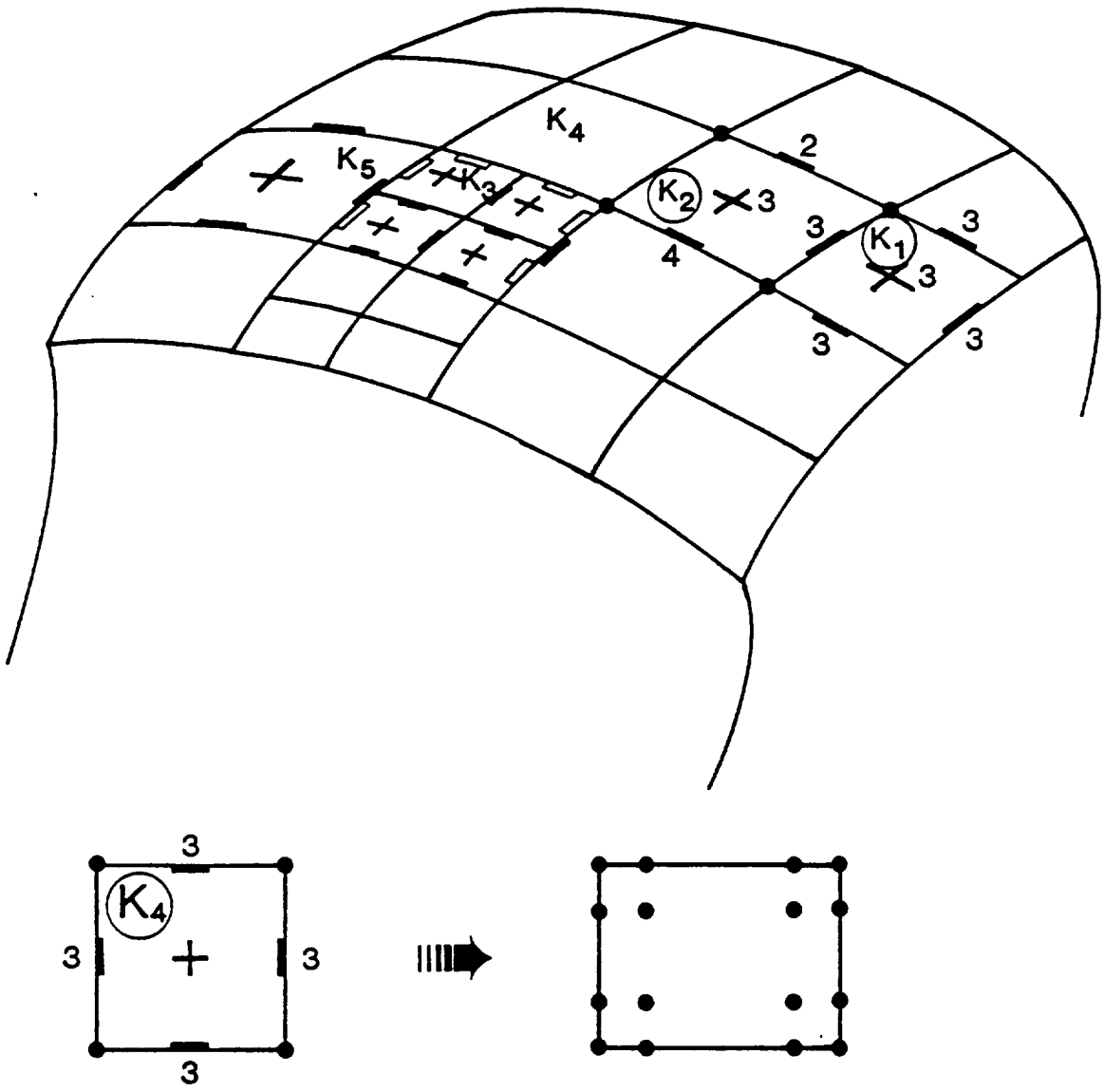


Figure 6.1: Sample *h-p* grid containing both active and constrained nodes.

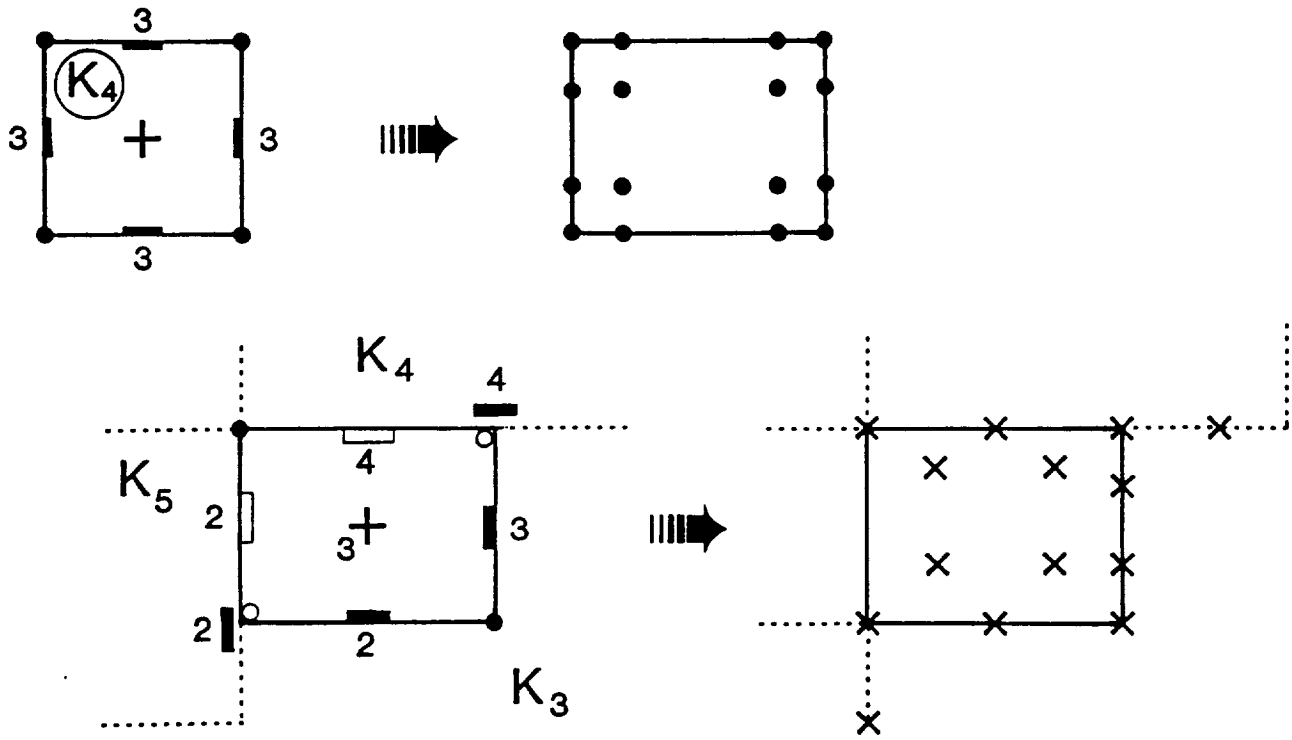


Figure 6.2: Integration point locations used to avoid locking.

The problem of enforcing mixed boundary conditions involves also a choice of approximation of the outward normal to the boundary. The simplest solution is to define the outward normal at a point common to several elements as an average of outward normals evaluated for all these elements.

6.2 Routines for Performing Refinements in Boundary Layer Zone

As mentioned in the section on adaptivity, p-refinements along the solid wall boundaries are currently not performed automatically, but rather "by hand." That is, a user decides which elements should be enriched. In fact, designing a mesh in the boundary layer zone may also require some other operations like breaking or unrefining some elements. If one solves a test problem with a relatively small number of elements, such an operation can be performed interactively. In the case of a real life problem, however, the user would have to indicate a large number of elements to be refined or unrefined.

A method has been developed that significantly reduces the effort of interactive generation (refinement and unrefinement) of simple meshes. The method consists of mapping a possibly curvilinear portion of a mesh into a rectangular domain with element sides parallel to the sides of a rectangle. Of course, the method can be applied only to pieces of the mesh for which the pattern of initial elements is topologically equivalent to a uniform rectangular mesh.

The idea of using the mapped image of the mesh to perform the required refinements is simple. First one identifies the rectangular coordinates of a group of elements that they wish to refine (this is an easy operation as mapped elements constitute rectangular patterns). Then one performs a specified refinement by giving the coordinates of rectangles covering the considered subdomain. For instance, one may prescribe: break elements between $x = 0$ and $x = 1$ and between $y = 0.5$ and $y = .75$, etc. This way of generating meshes and simple refinements is still interactive yet it allows one to perform massive refinements with a minimal effort. (See the User Manual for additional details.)

6.3 Postprocessing in Three Dimensions

When solving three–dimensional problems, special attention must be devoted to the problem of displaying the solution. The three–dimensional finite element code is equipped with a postprocessing package which can display contour maps of the solution on the boundary and on the desired cross sections of the computational domain. This postprocessing is rather expensive and it is not always sufficient for displaying interesting features of the solution.

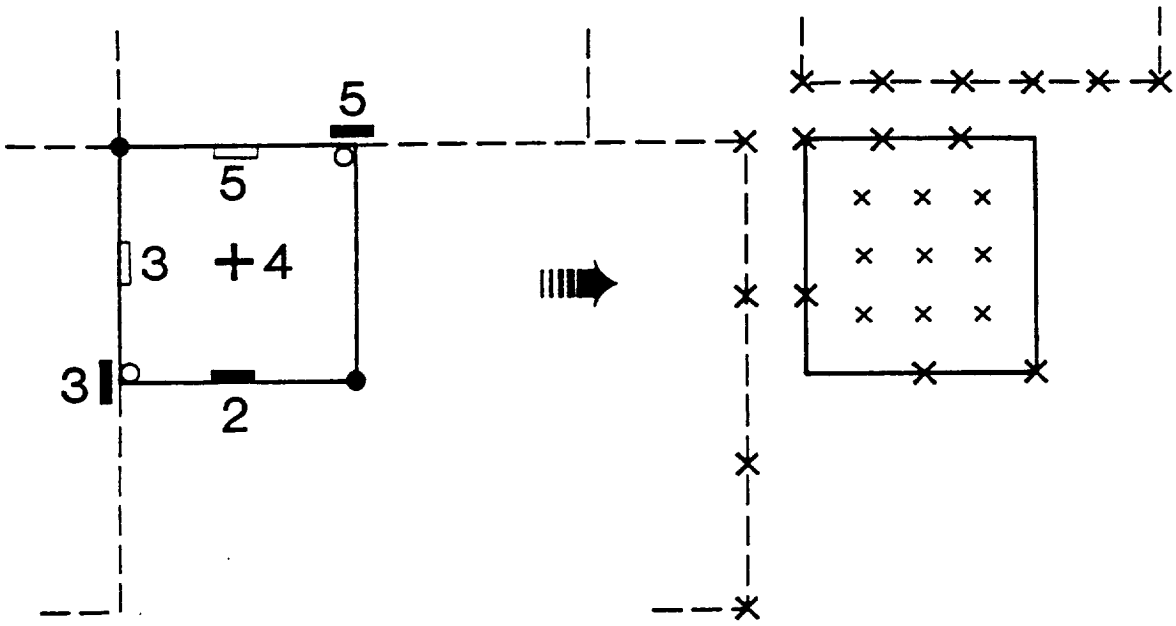


Figure 6.3: Sampling points for integrating penalty terms.

For instance, contour maps give only a rather inaccurate image of quantities defined on the boundary of the domain such as, say, heat flux or skin friction. Three–dimensional perspectives of such functions give a much better idea of their behavior.

Following this motivation an algorithm was developed for displaying three–dimensional perspectives of the solution dependent functions on the portions of the boundary of a three–dimensional domain. In the first step, the indicated portion of the boundary, possibly curvilinear, is continuously mapped into a flat plane. Two kinds of mappings are possible; a projection in a given direction or the mapping "stretching" a two–dimensional mesh of distorted rectangles on the boundary into a plain mesh of distorted rectangles on the boundary into a plain mesh of rectangles, the transformation described in the previous section. In the second step, a three–dimensional perspective of the solution is displayed over the plane domain.

The procedure is inexpensive if compared to contouring in three dimensions and it is very helpful in showing intricate features of a three–dimensional solution.

6.4 Solution Correcting Procedures

When integrating the Euler or Navier-Stokes equations one frequently encounters a cumberson phenomenon: the solution evolves to a physically unacceptable state with negative densities or pressures. Such situations make further integration impossible unless one artificially corrects the solution by "pushing it back" to physically meaningful values.

A procedure for performing this operation was designed. It can be outlined as follows: First, one verifies if

$$\begin{cases} \rho > 0 \\ \iota = e - \frac{1}{2}\left(m_1^2 + m_2^2\right)/\rho \geq 0 \end{cases}$$

where ρ is density, ι internal energy, m, n momentum components, and e is the total energy. Note that the above conditions imply that $e \geq 0$ and the pressure $p = (\gamma - 1)\iota > 0$. If either $\rho \leq 0$ or $\iota < 0$, one replaces ρ and e by new values ρ^* and e^* defined as a projection of the point $(\rho, e) \in \mathbb{R}^2$ onto a set $A \subset \mathbb{R}^2$ of physically acceptable densities and energies:

$$A = \left\{(\rho, e) \in \mathbb{R}^2 : \rho \geq 0 \text{ and } \rho e \geq \frac{1}{2}\left(m^2 + n^2\right)\right\}$$

The correcting procedure described here has proven very useful especially when starting the integration process when the initial solution may not satisfy given boundary conditions and lead immediately to nonphysical states.

7 Numerical Examples

Several problems were solved to test the algorithms presented in the previous sections. The problems range from the simplest two–dimensional inviscid flow over a wedge to three–dimensional viscous flows with high Mach and Reynolds numbers. In all of the numerical examples presented here we have focused on providing solutions exclusively for steady state problems. The integration in time process is a method of obtaining such solutions. After converging to steady state, error estimation and mesh adaptation are performed according to rules presented in Section 4. The residual error indicators are used as the basis for mesh refinement with the equidistribution of errors as a refinement strategy.

The stable time step Δt used in the calculations has been evaluated by the following formula

$$\Delta t = CFL \min_K \left| \frac{h_K}{|\boldsymbol{u}| + c} \right| \tag{7.1}$$

where $|\boldsymbol{u}|$ is the maximum magnitude of the fluid velocity in the element K, c is the speed of sound, and CFL is the Courant, Friedrichs, Lewy stability coefficient [6]. In addition, all three types of artificial dissipation, the Lapidus and Morgan's viscosities and a generalization of the last one for distorted elements have been used for various problems. The implicitness parameters β and γ in all the examples are set equal to one. As a linear equation solver we have used the block Jacobi algorithm accelerted with GMRES.

Example 1: Inviscid Flow Over a Wedge on h-Adapted Meshes

We consider the problem of supersonic flow over a wedge with the following prescribed data:

$$\text{Mach number } M = 3.0$$

$$\text{Inclination of the wedge } \theta = 20°$$

The initial meshes consist of 12×5 linear, quadratic, and cubic elements. The Lapidus viscosity constants for these three cases were assumed as $c_k = 1$, $c_k = 0.15$, $c_k = 0.07$, respectively.

The h-adaptive meshes are shown in Figs. 7.1, 7.3, and 7.5, and the corresponding density contours are given in Figs. 7.2, 7.4, and 7.6. Comparing these figures one finds that all three meshes have captured the shock within two or three elements and all of the shock angles are virtually identical. In addition, the density contours obtained from the linear mesh (Fig. 7.2) show a much tighter shock pattern than the quadratic or cubic meshes (Figs. 7.4 and 7.6). This is most likely a result of the two additional levels of h-refinement and an introduction of approximately 50 percent more degrees of freedom in the shock region than in either of

the other two cases. Finally, note that all these cases show very little of any reflection of the shock at the subsonic outflow boundary on the upper surface.

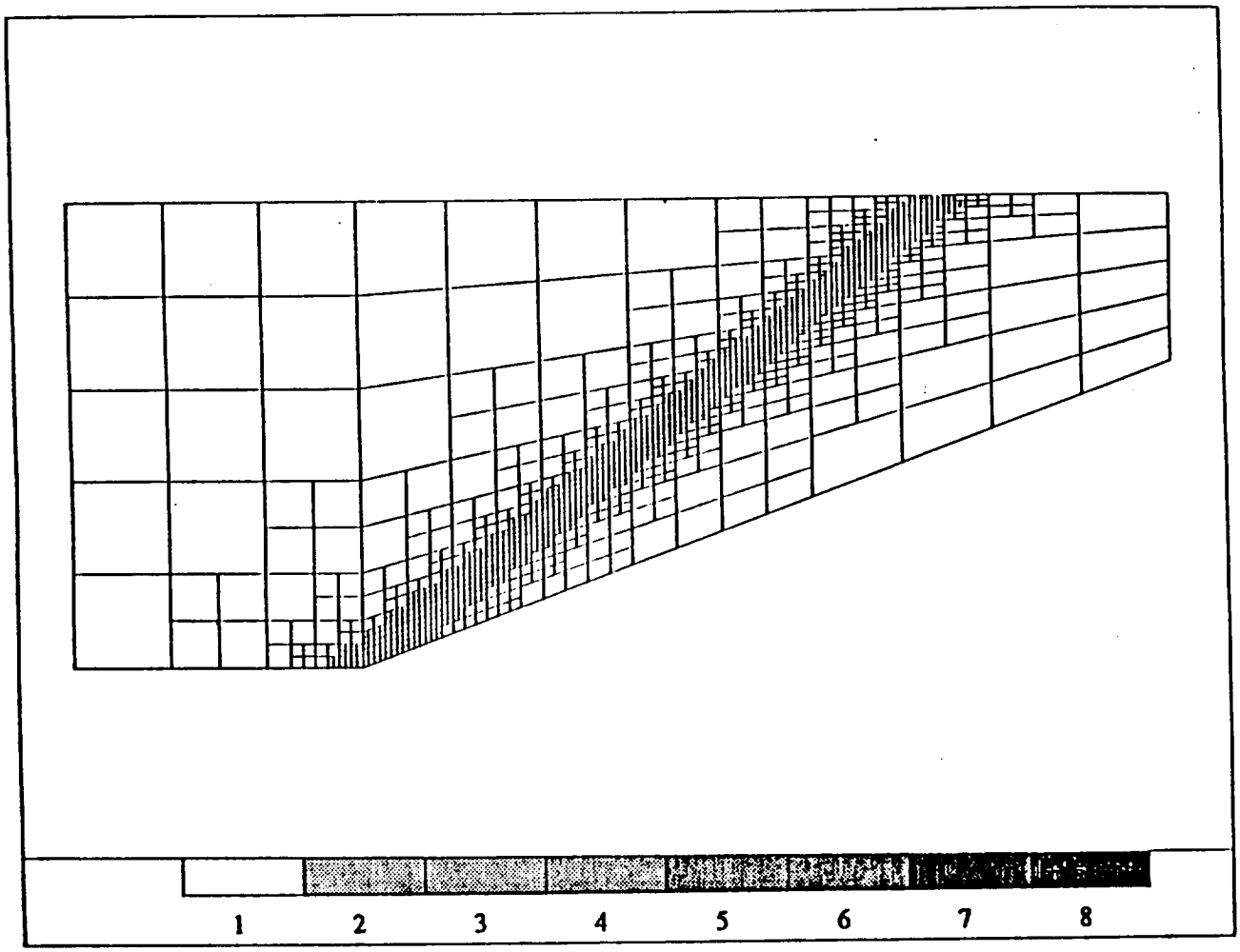


Figure 7.1: Flow over a wedge problem on an h-adaptive mesh of linear elements. Final mesh.

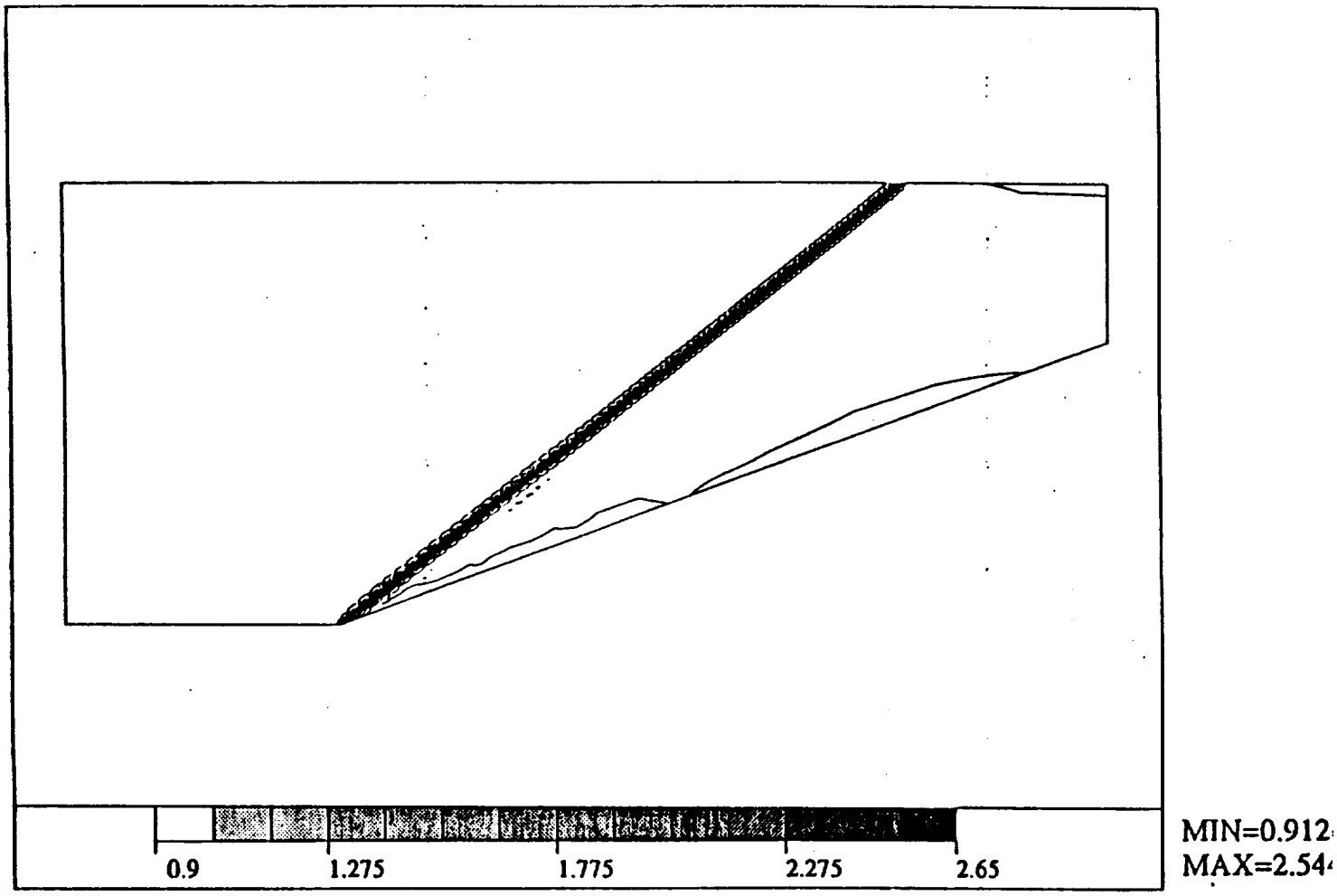


Figure 7.2: Flow over a wedge problem on an h-adaptive mesh of linear elements. Density contours.

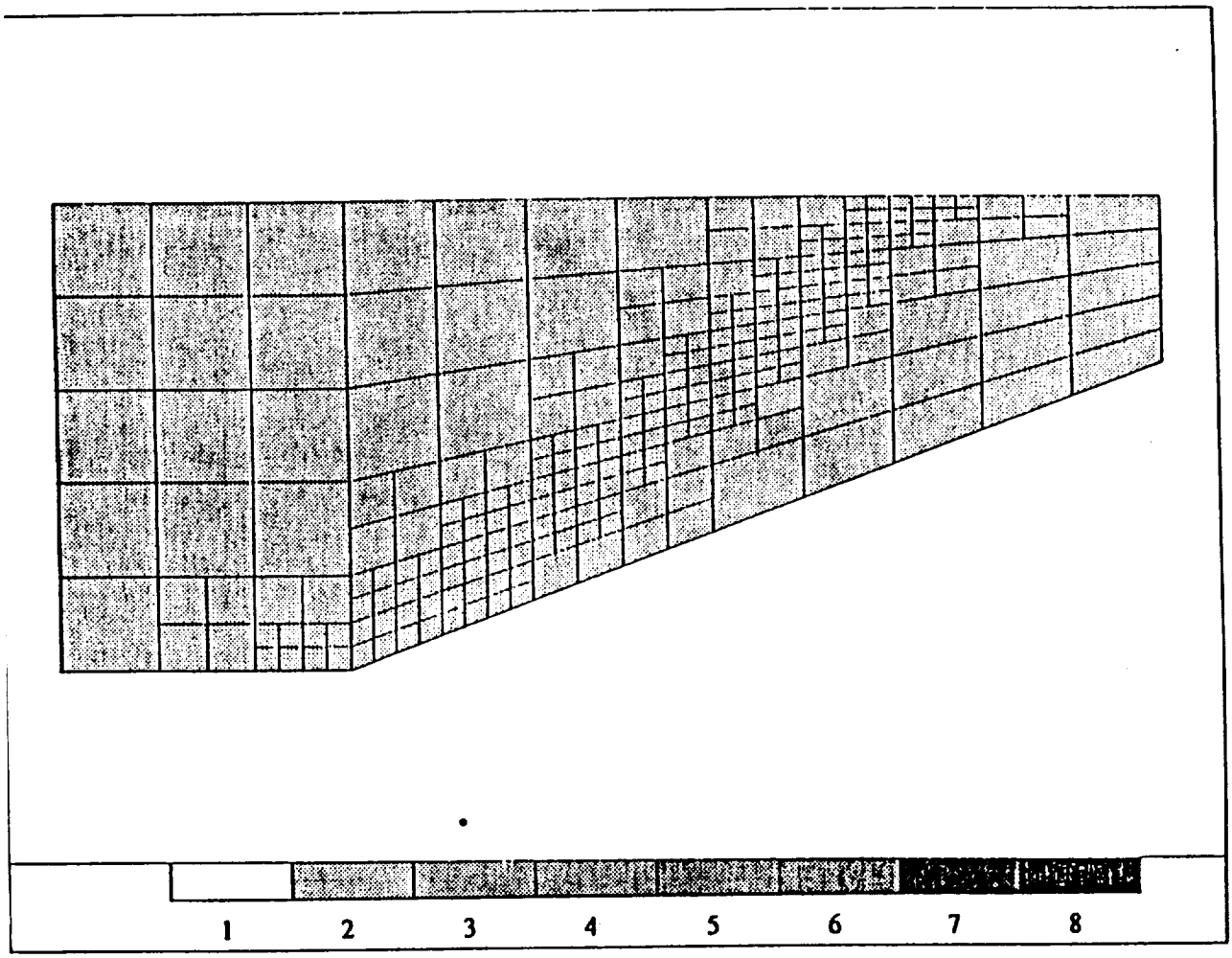


Figure 7.3: Flow over a wedge problem on an h-adaptive mesh of quadratic elements. Mesh after two levels of refinements.

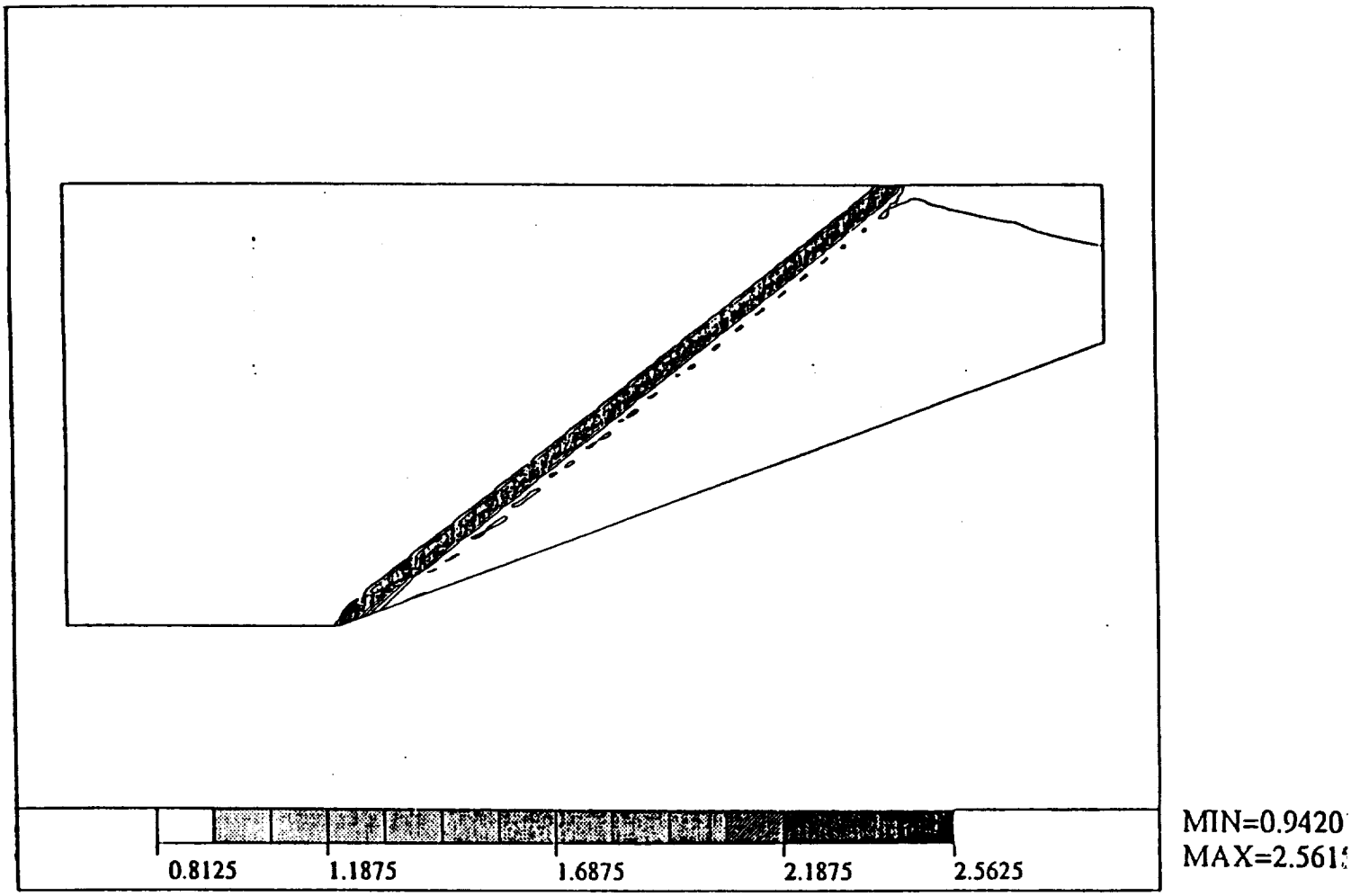


Figure 7.4: Flow over a wedge problem on an h-adaptive mesh of quadratic elements. Density contours.

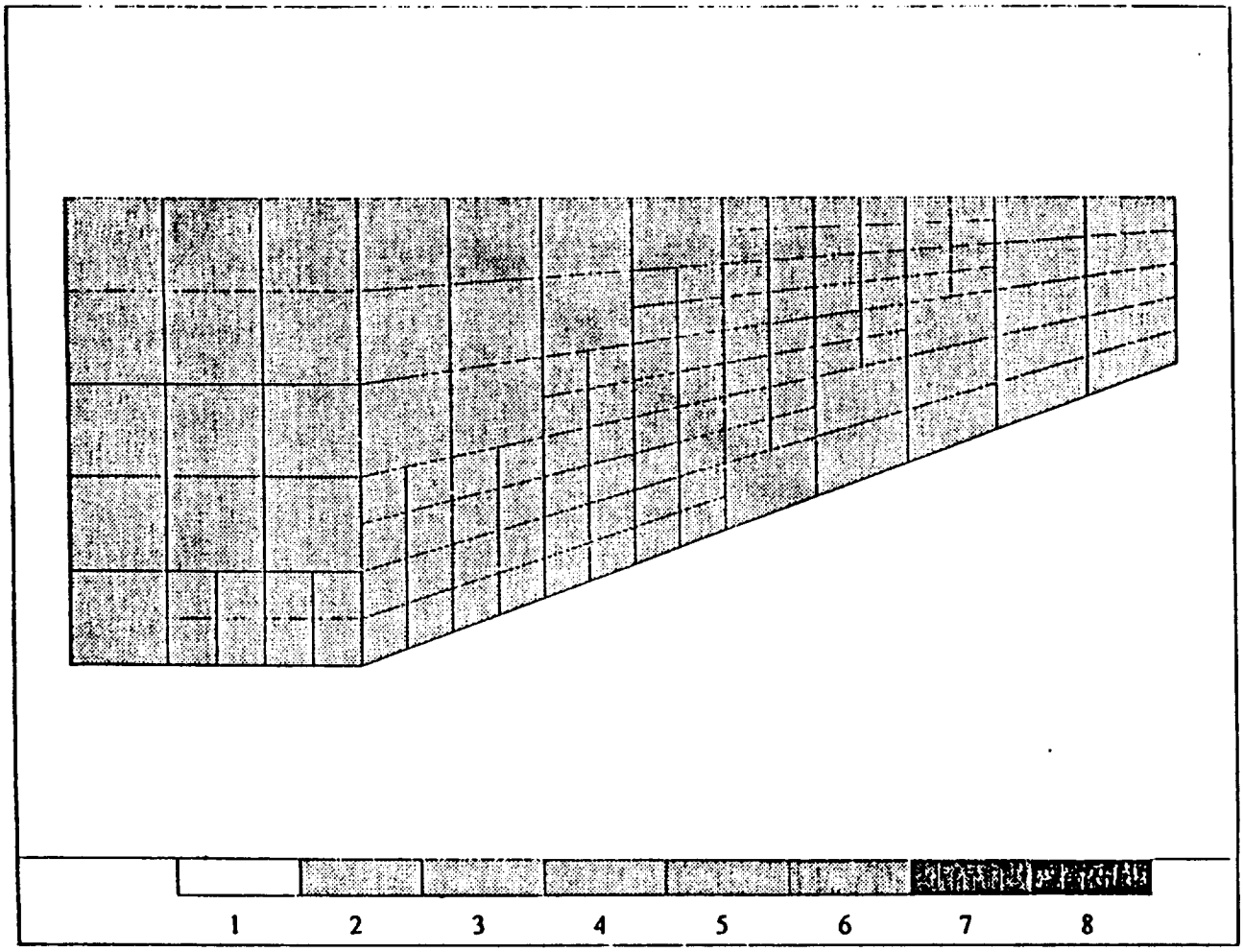


Figure 7.5: Flow over a wedge problem on an h-adaptive mesh of cubic elements. Mesh after one level of refinement.

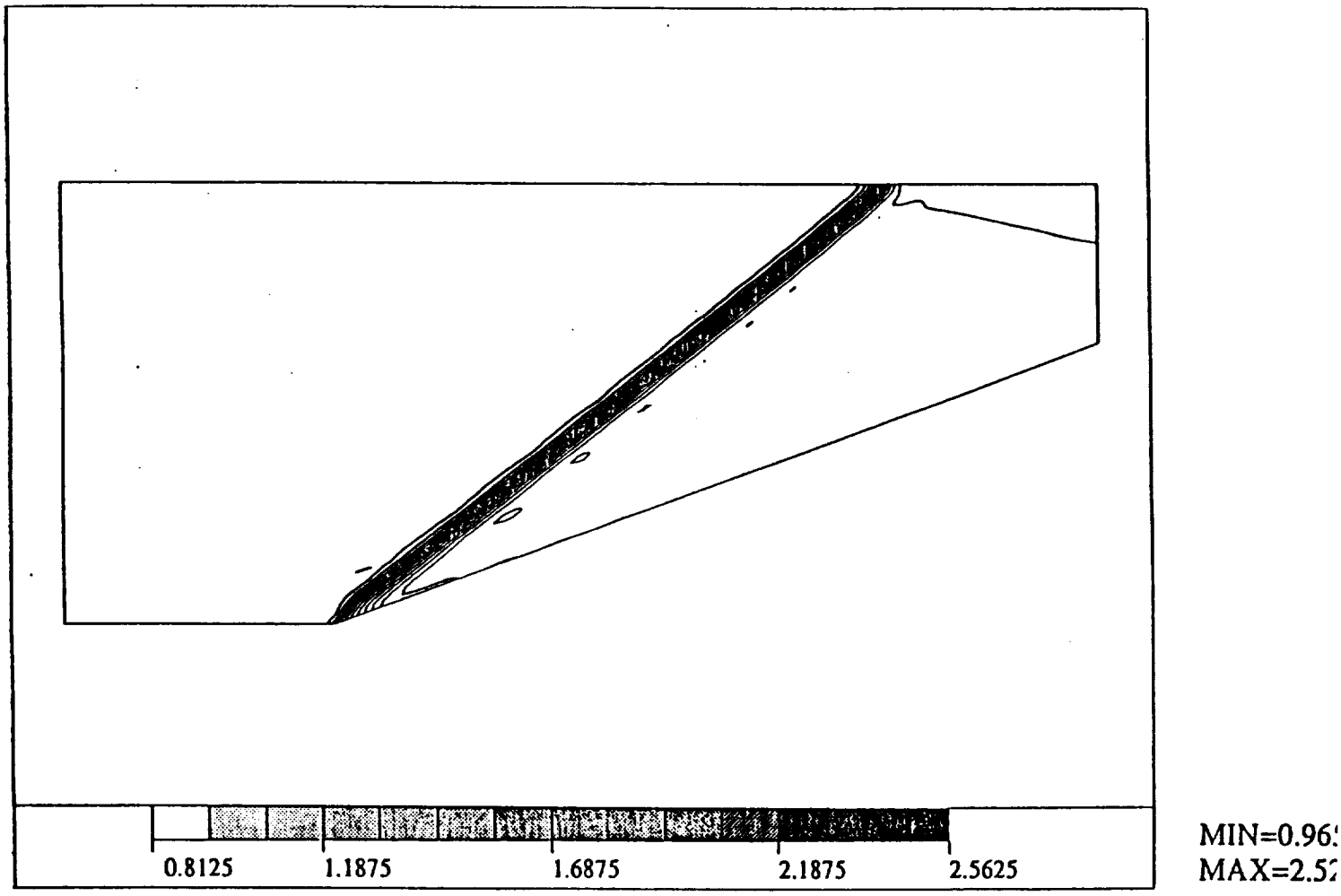


Figure 7.6: Flow over a wedge problems on an h-adaptive mesh of cubic elements. Density contours.

Example 2: Inviscid Flow Over a Blunt Body on h-Adapted Meshes

As a second test problem, we considered the supersonic flow over a cylinder with Mach number $M = 6.0$. The problem was solved starting with initial meshes of 16×16 linear elements and 8×8 quadratic and cubic elements. The viscosity constants c_k were set to 1.0, 0.07, and 0.035, respectively. The final h-adapted meshes are presented in Figs. 7.7, 7.9, and 7.11, the corresponding density contours are shown in Figs. 7.8, 7.10, and 7.12. Comparing these results one finds again that all three meshes have predicted virtually identical stand–off distances for the bow shock, and have captured the shock within two or three elements. The resolution of the shock for the linear and quadratic meshes in this case is again somewhat better than for the cubic mesh, which is most likely due to the number of degrees of freedom introduced in the shock region. Finally, note that the density contours for the cubic elements are the smoothest of the three meshes. This same pattern can be observed in Figs. 7.2, 7.4, and 7.6 of Example 1.

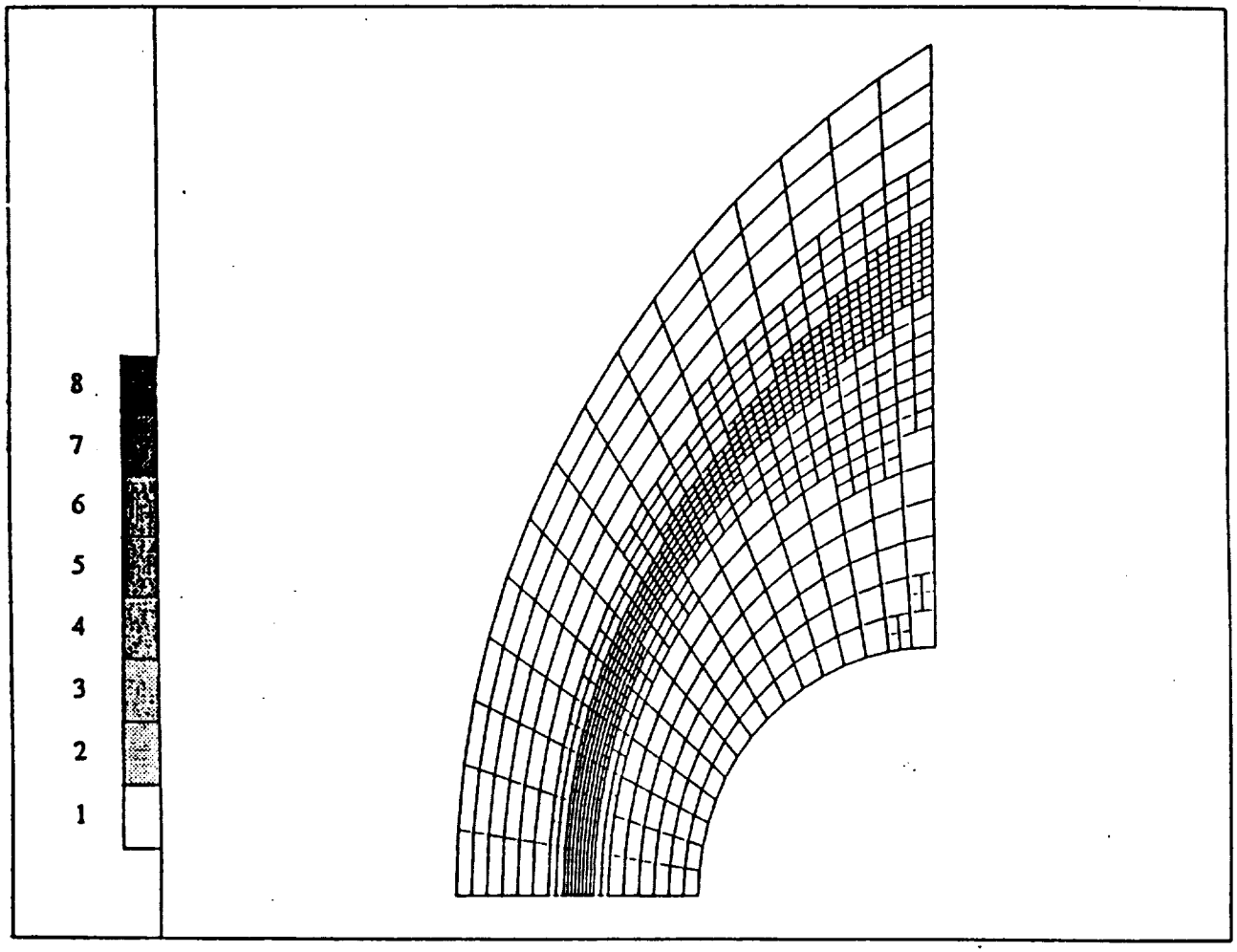


Figure 7.7: Blunt body problem on an h-adaptive mesh of linear elements. Mesh after two levels of refinements.

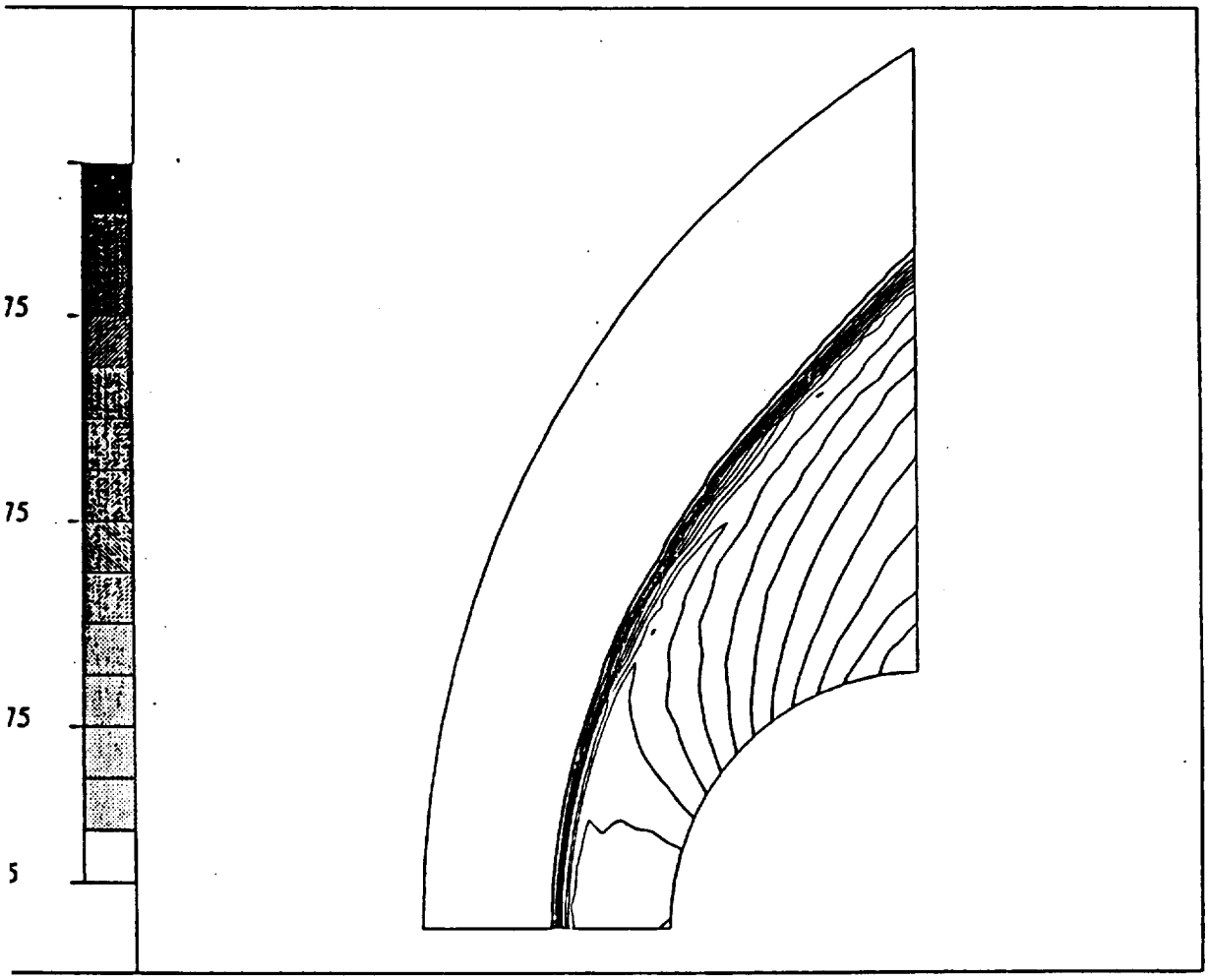


Figure 7.8: Blunt body problem on an h-adaptive mesh of linear elements. Density contours.

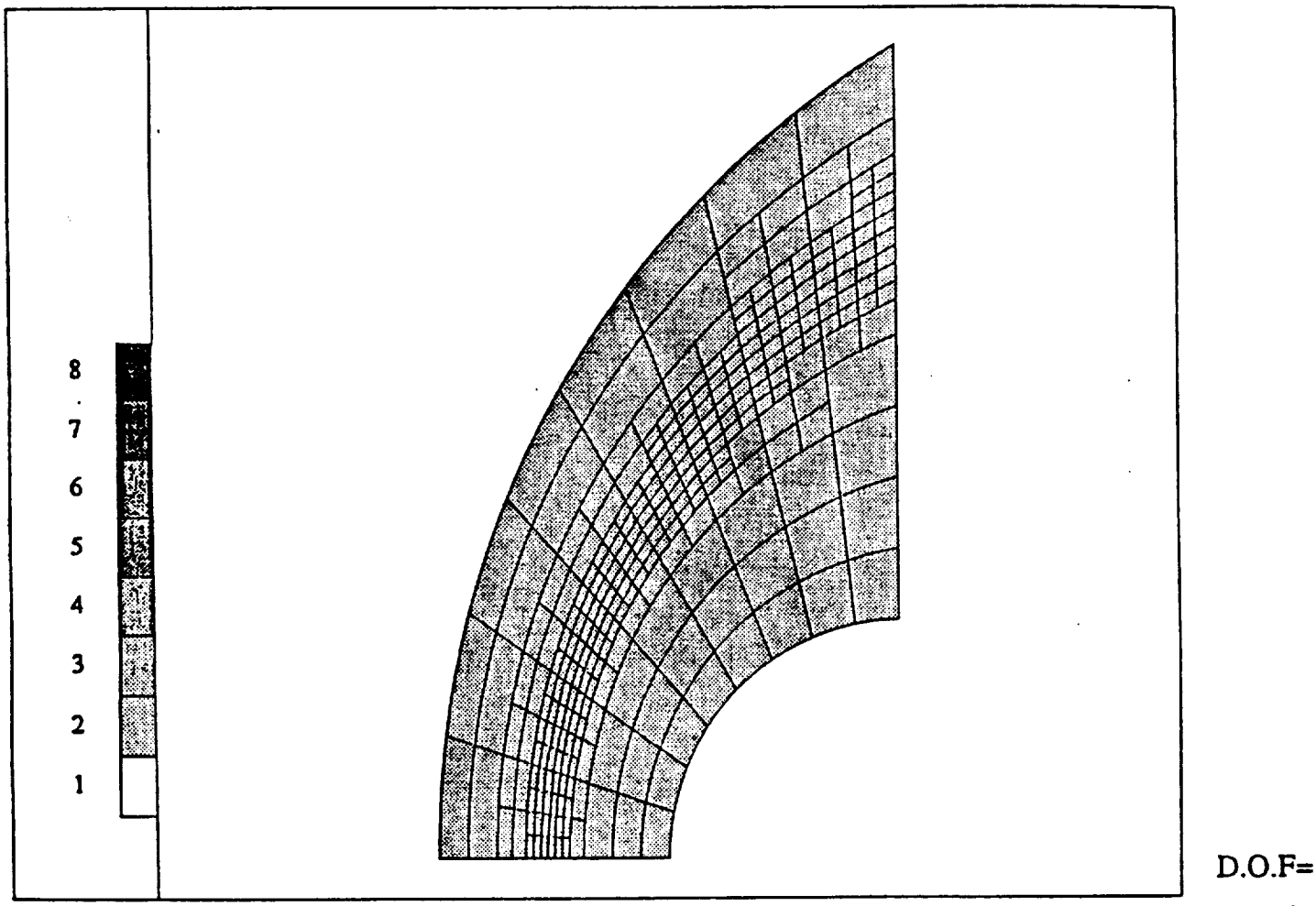


Figure 7.9: Blunt body problem on an h-adaptive mesh of quadratic elements. Mesh with two levels of refinements.

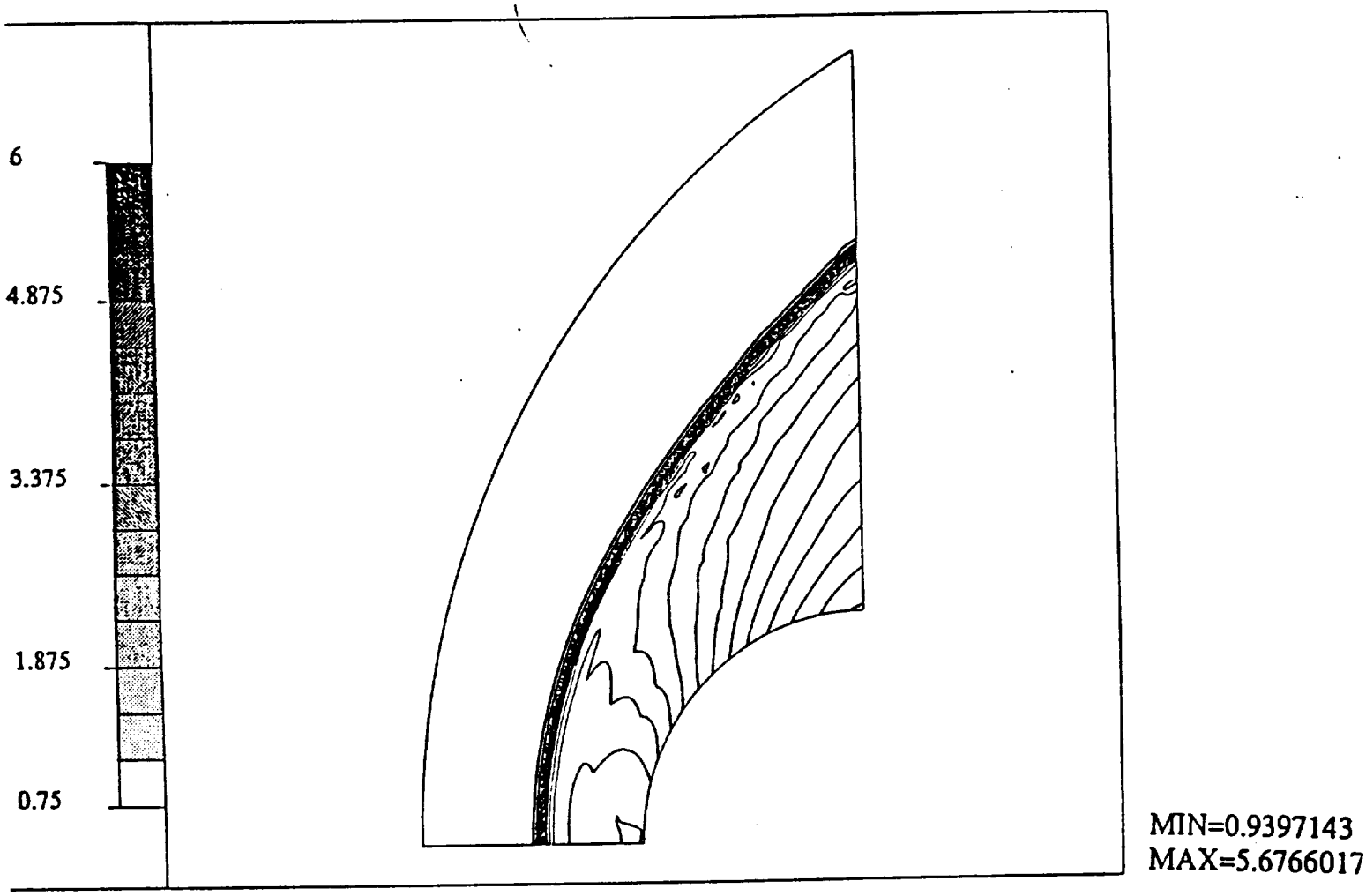


Figure 7.10: Blunt body problem on an h-adaptive mesh of quadratic elements. Density contours.

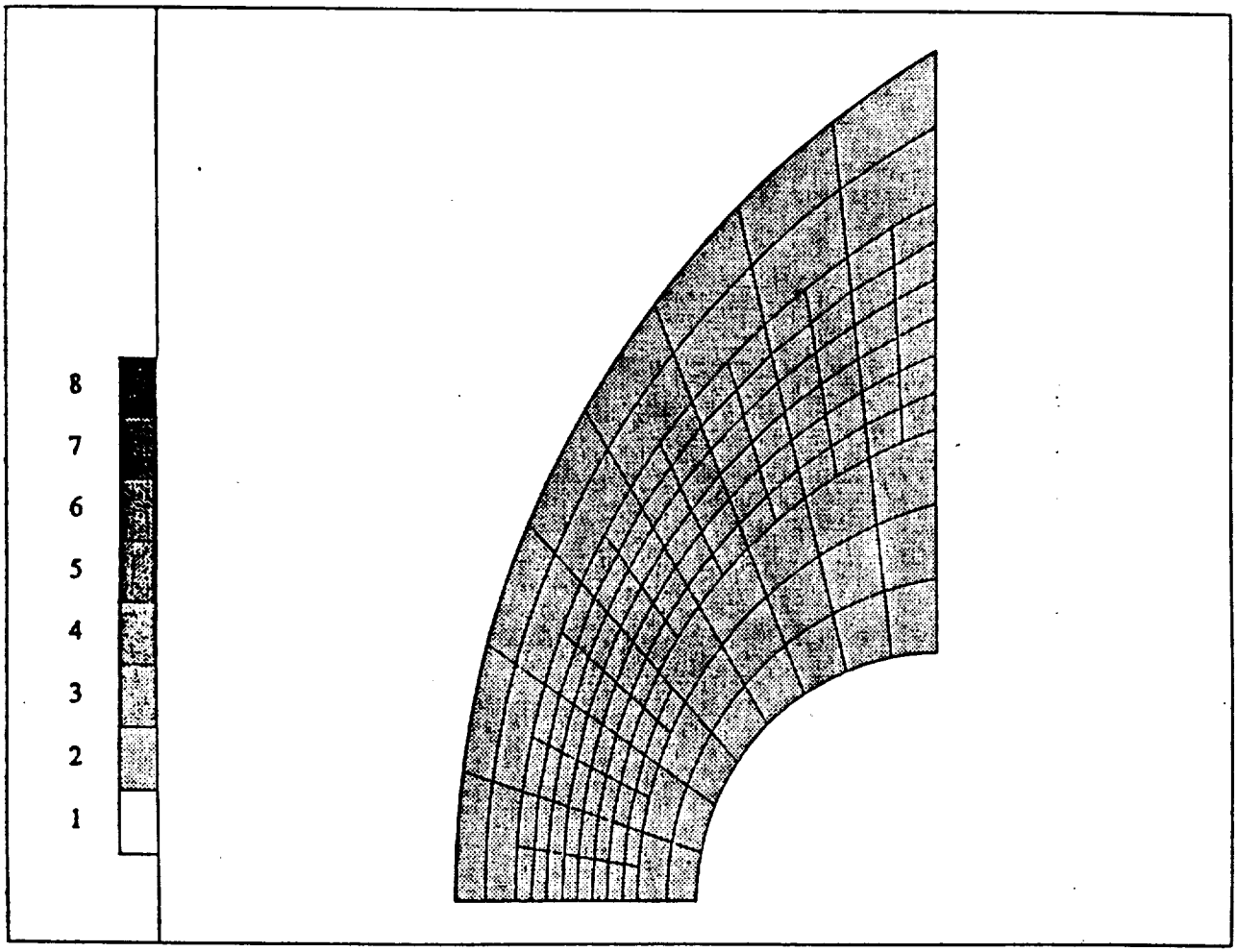


Figure 7.11: Blunt body problem on an h-adaptive mesh of cubic elements. Mesh after one level of refinements.

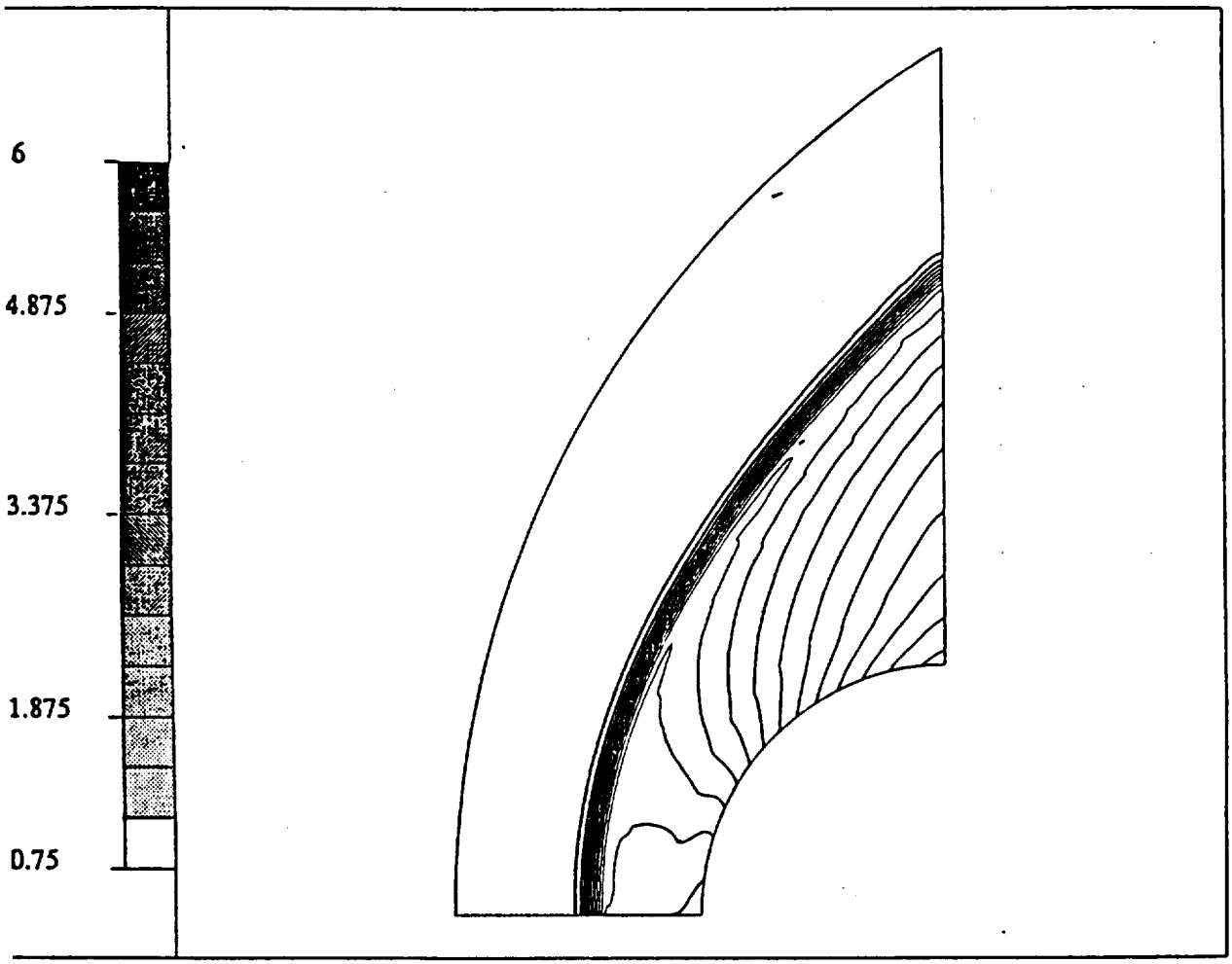


Figure 7.12: Blunt body problem on an h-adaptive mesh of cubic elements. Density contours.

Example 3: Carter's Flat Plate Problem

The third test case we considered was Carter's Flat Plate Problem [5] with the following data

$$M_\infty = 3, \quad Re_L = 1000, \quad Pr = 0.72, \quad \gamma = 1.4, \quad T_\infty = 390[^\circ R] \tag{7.2}$$

The geometry and boundary conditions are shown in Fig. 7.13. The following flow featues are recognized:

- a singularity exists near the leading edge of the plate,
- a curved bow shock is developed from the tip of the plate, and
- a boundary layer is formed along the plate.

The flags in the figure (KIND = 1,..., 5) correspond to various boundary conditions incorporated in the code. The solid wall temperature is

$$T_{\text{wall}} = 1092[^\circ R] \tag{7.3}$$

A nonuniform initial h-p finite element mesh, as shown in Fig. 7.14 with second and third order elements defined along the solid wall boundary, has been used to march to the steady state condition. In order to reduce the numbers of degrees of freedom in most of the boundary layer region (except near the stagnation point), anisotropic elements were used which have a high order of approximation in the direction normal to the plate and a linear approximation along the plate.

Using the pure h-adaptive strategy described in Section 4, the mesh shown in Fig. 7.15 was obtained. The corresponding density contours are presented in Fig. 7.16. The next two sequences of adaptive meshes are shown in Figs. 7.17 and 7.19 with the corresponding density contours in Figs. 7.18 and 7.20. After the third adaptive refinement, the solution process was stopped when the error reached the prescribed error tolerance. In all steps, the CFL constant was set to 0.5.

Figures 7.21 and 7.22 compare the computed profiles of pressure and skin friction distributions along the solid wall with the original results of Carter, showing good agreement away from the stagnation point and the ability of the method to also resolve the stagnation point itself. Figure 7.23 shows the computed heat flux along the solid wall (not presented in Carter's report). Note the positive sign in the vicinity of the tip of the plate (the gas heating plate). The stabilizing effect of higher order elements near the solid wall can also be observed in the pressure contours shown in Fig. 7.24 corresponding to the final mesh.

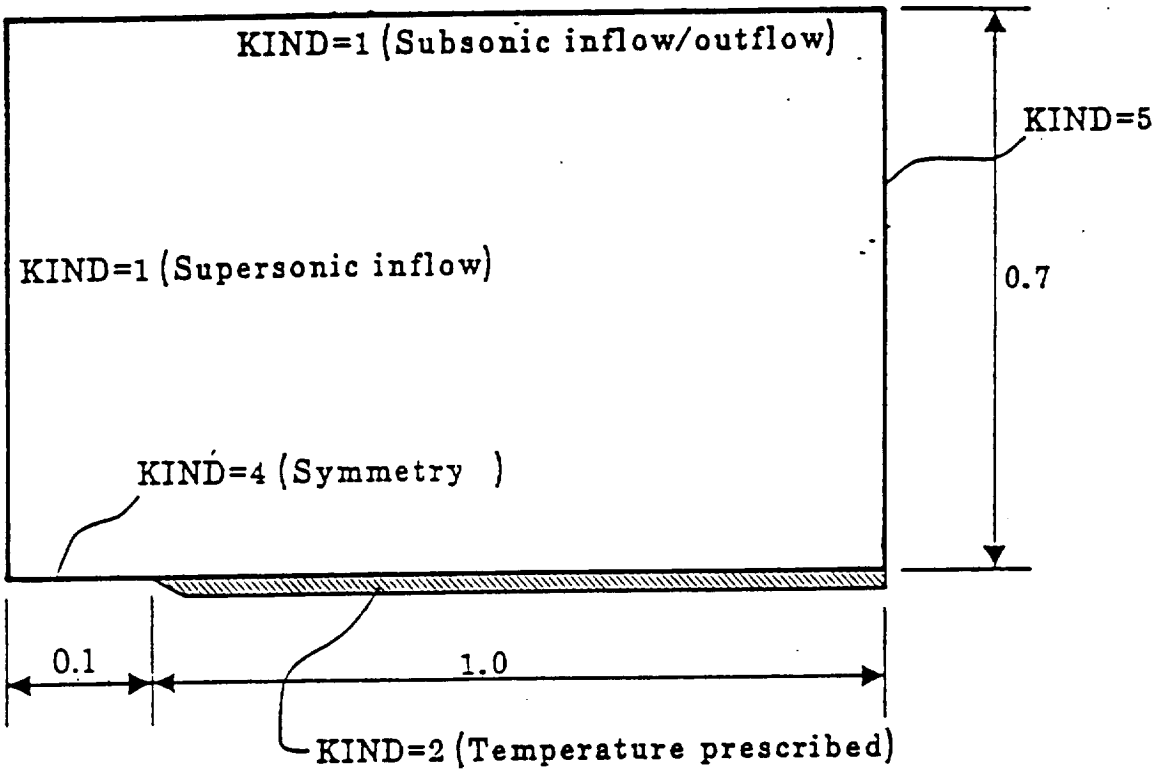


Figure 7.13: Carter's flat plate problem. Geometry and boundary conditions.

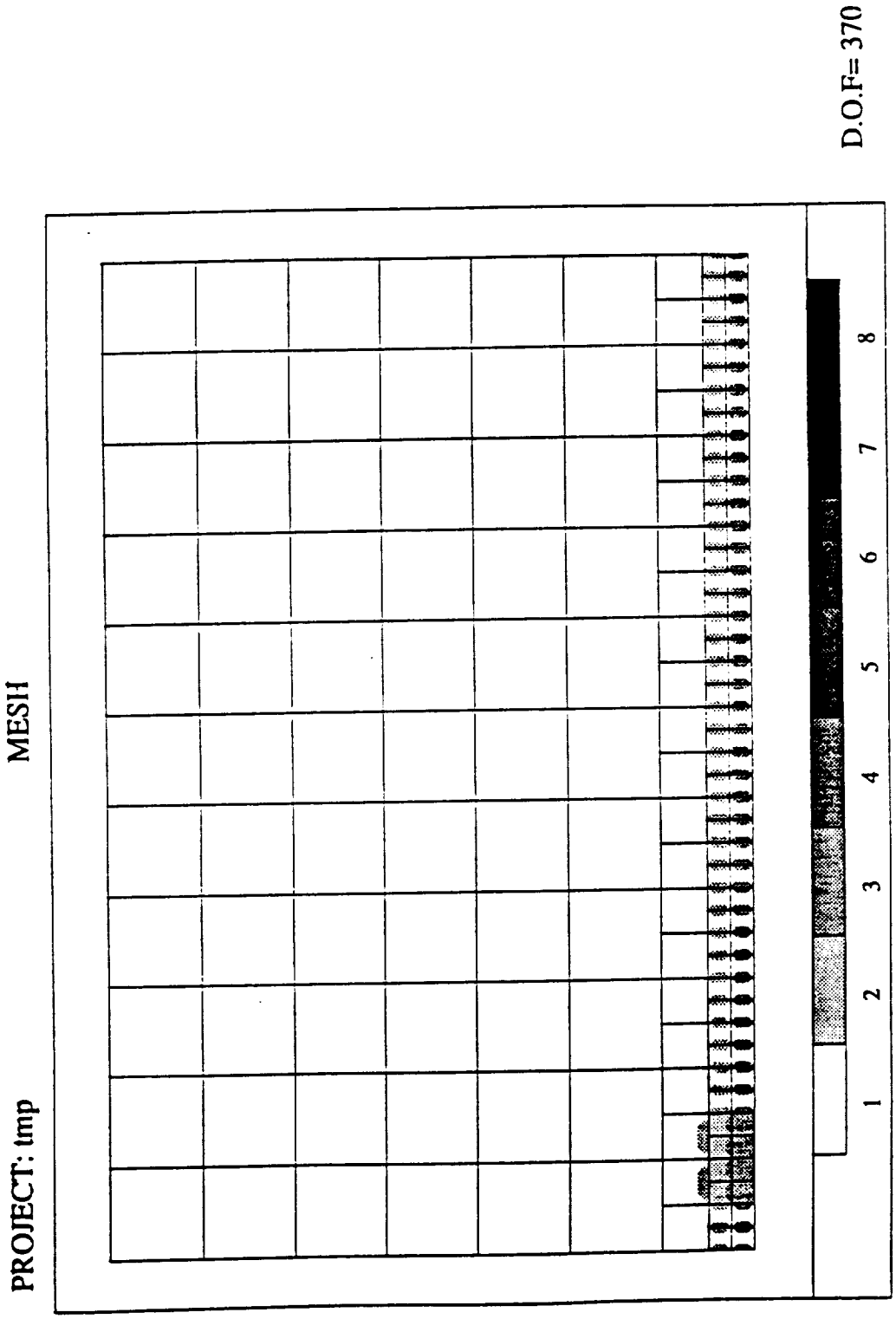


Figure 7.14: Carter's flat plate problem. A nonuniform initial h-p mesh.

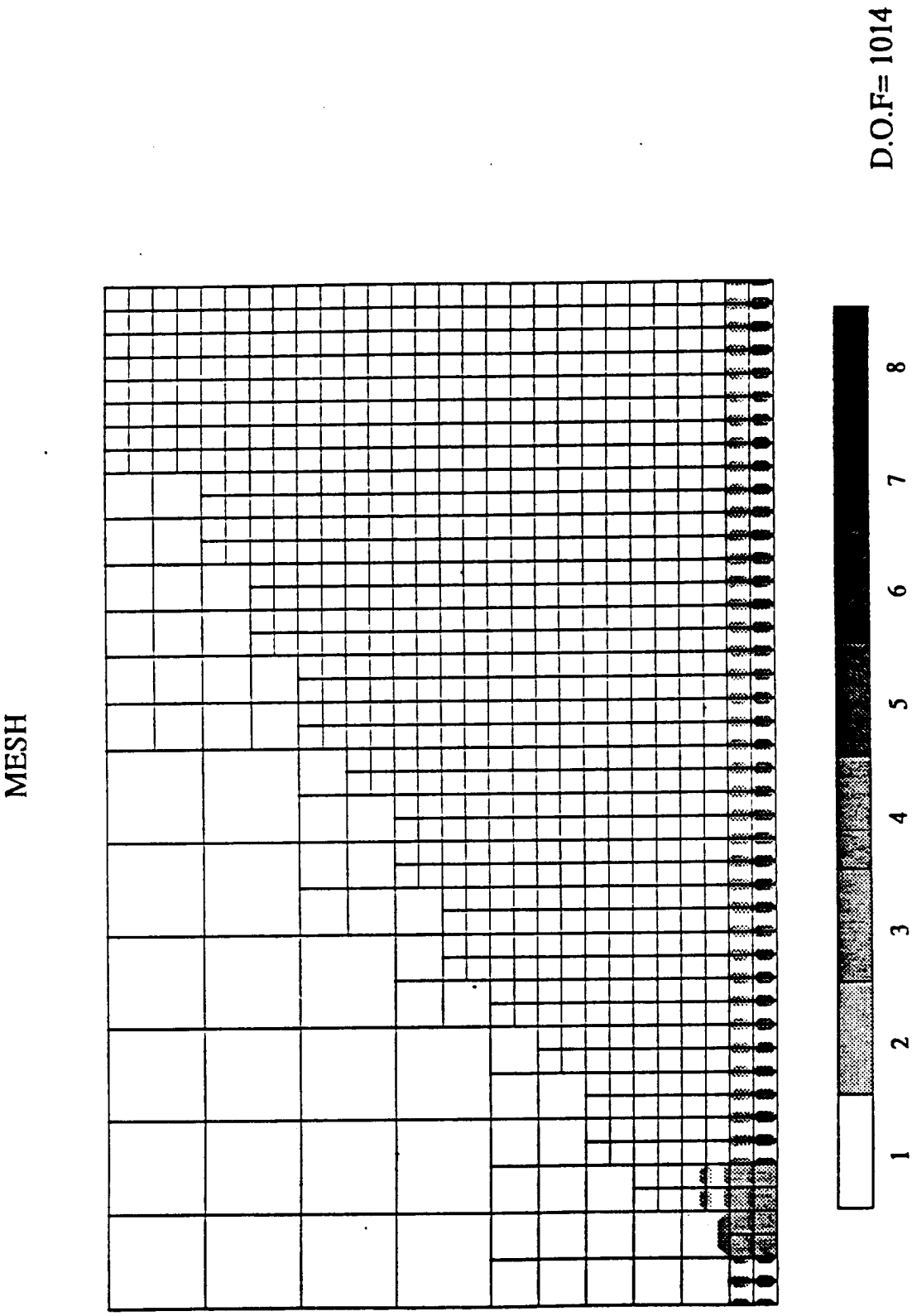


Figure 7.15: Carter's flat plate problem. An optimal mesh with maximum level of refinement equal 2 (Mesh 1).

Figure 7.16: Carter's problem. Mesh 1. Density contours.

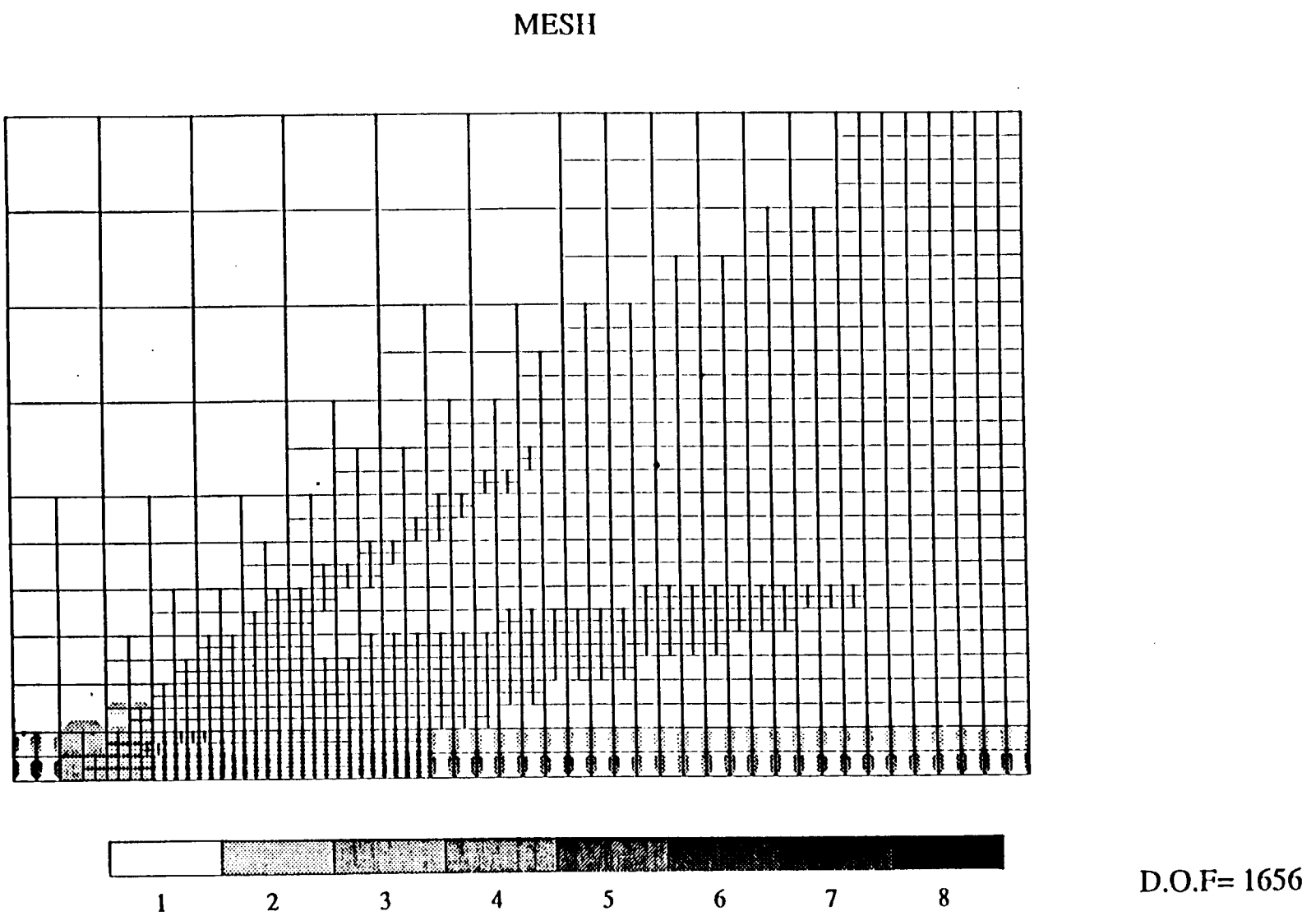


Figure 7.17: Carter's problem. Mesh 2.

Figure 7.18: Carter's problem. Mesh 2. Density contours.

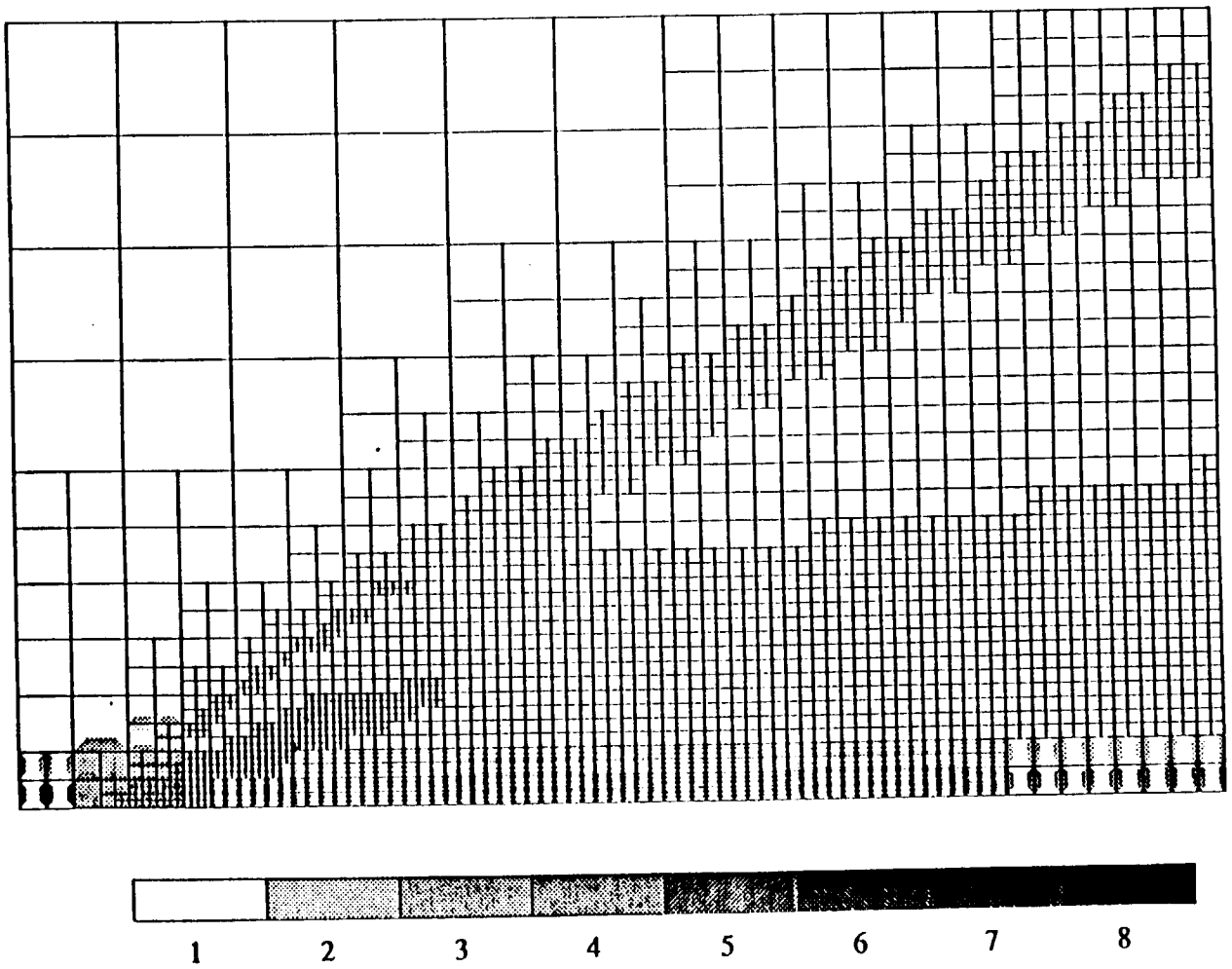


Figure 7.19: Carter's problem. Mesh 3.

Figure 7.20: Carter's problem. Mesh 3. Density contours.

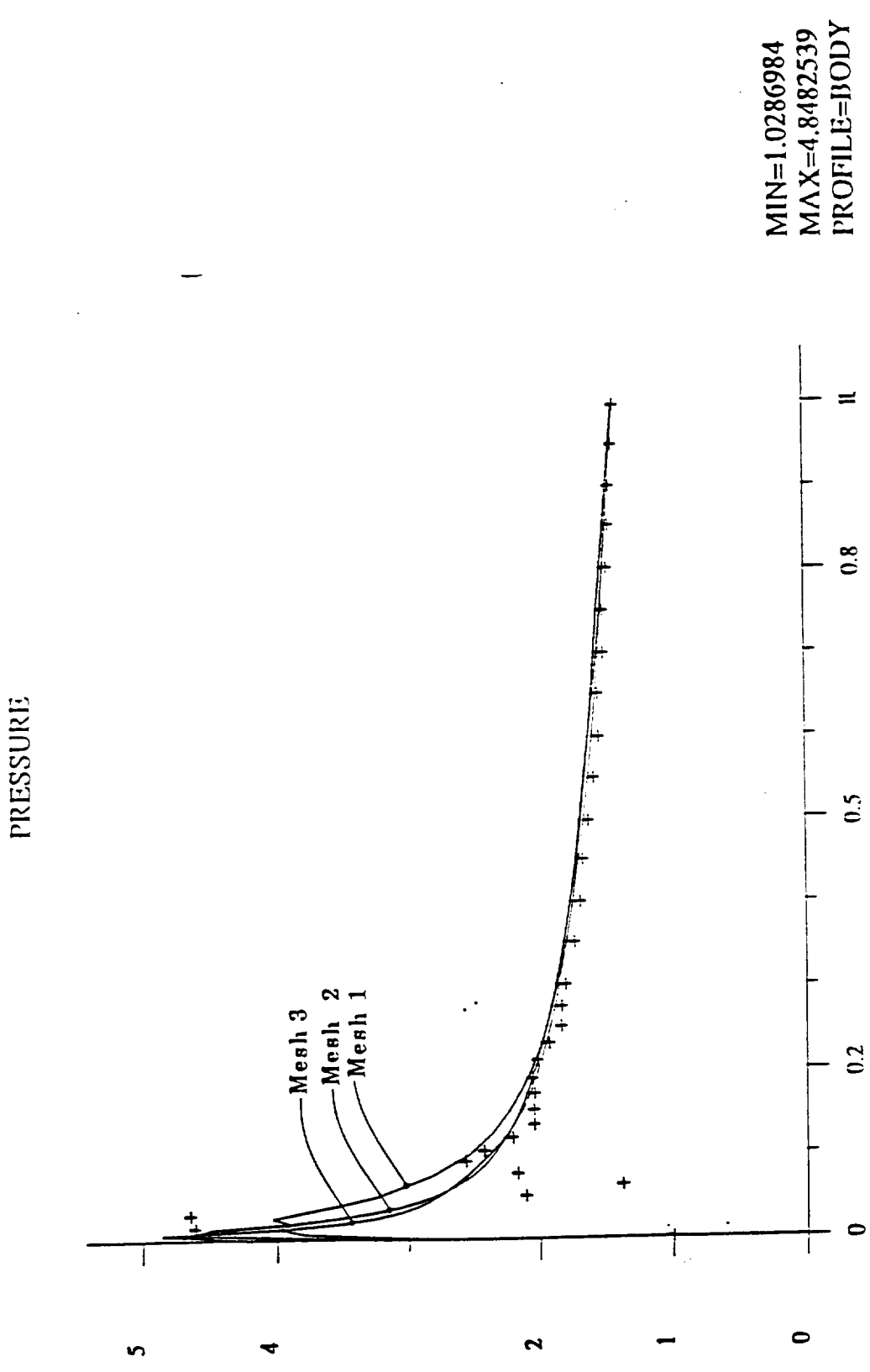


Figure 7.21: Carter's problem. Pressure profiles along the plate.

Figure 7.22: Carter's problem. Skin friction profiles along the plate.

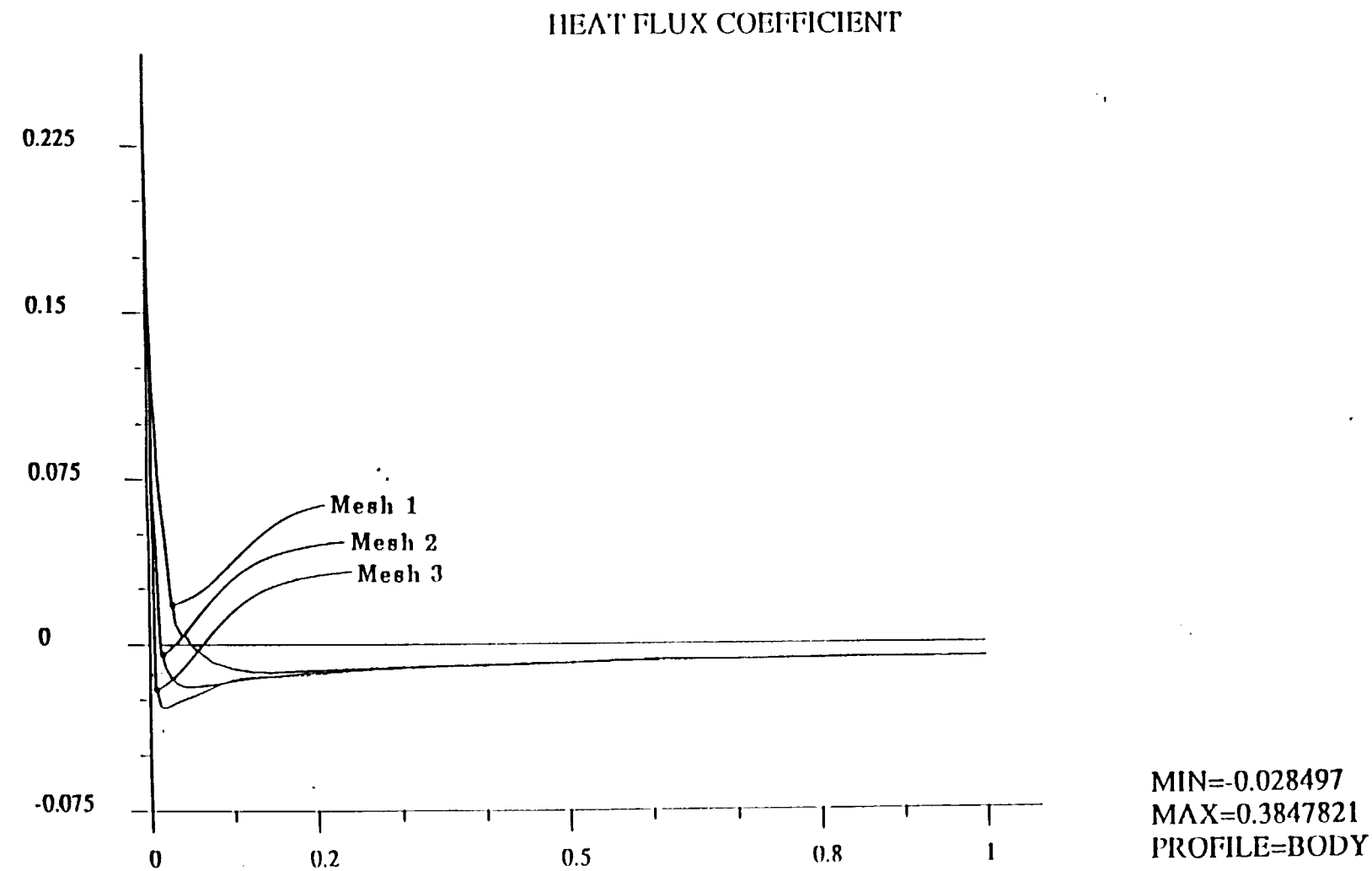


Figure 7.23: Carter's problem. Heat flux coefficient along the plate.

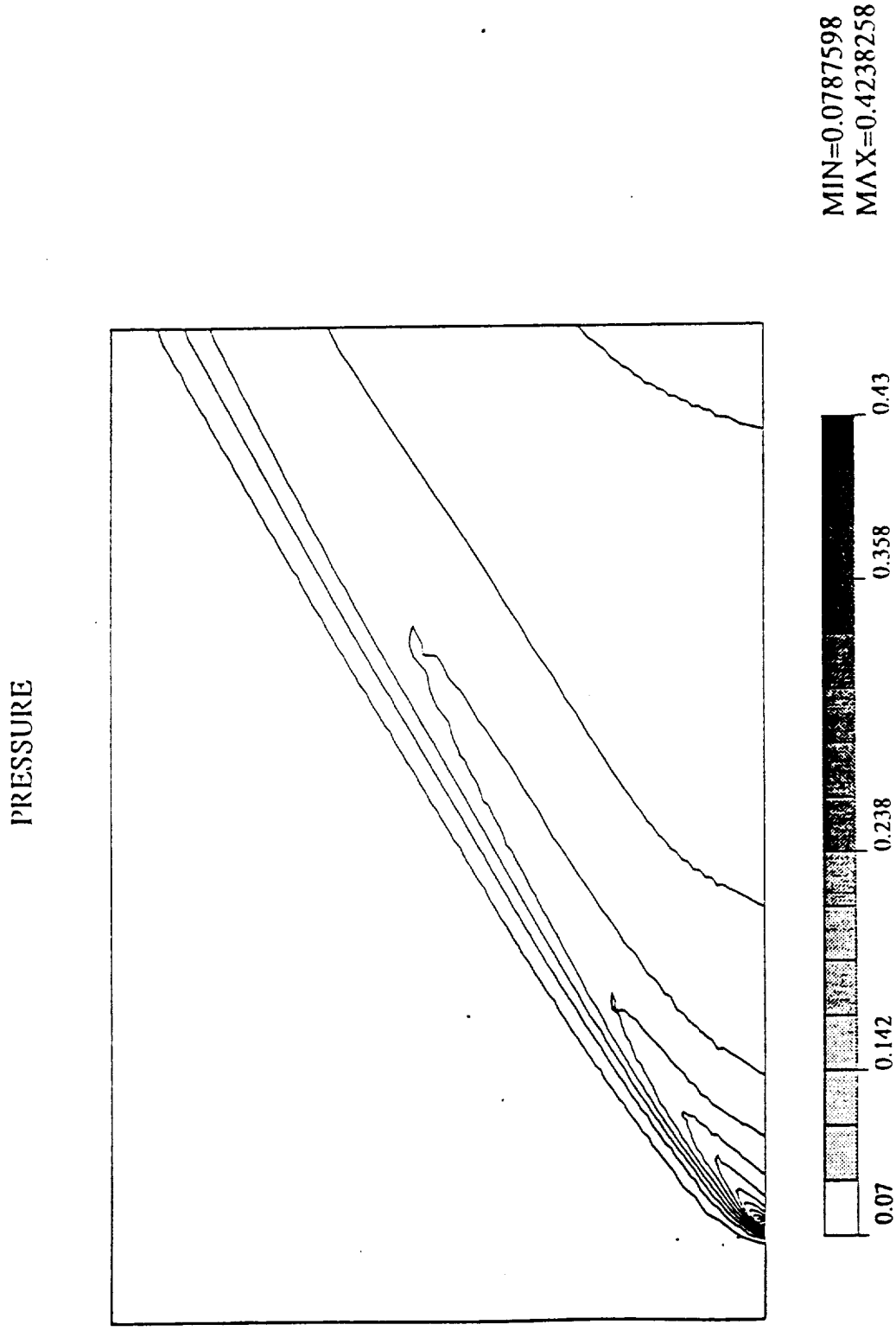


Figure 7.24: Carter's problem. Mesh 3. Pressure contours.

Example 4: Flat Plate Problem with $Re = 10,000$

As a continuation of Example 3, we have also performed an analysis of the flat plate problem with a Reynolds number of one order of magnitude larger. This required solving example problem 3 in a domain roughly ten times larger than before. In order to avoid an excessive number of degrees of freedom in the initial meshes, resulting from higher order elements in the initial stage of the solution procedure, higher order elements were introduced only in the last adaptive step after all h-refinements were performed.

Starting with an initial mesh of only 4×8 elements, a sequence of consecutive optimal linear meshes were obtained using the adaptive strategy discussed in Section 4. A total of 7 refinements were generated, each computed only when the optimal mesh for a particular level of refinement had been obtained. The final mesh and the corresponding density contours are shown in Fig. 7.25. Figure 7.26 presents a three–dimensional perspective of the same density function showing a clear separation of the shock from the boundary layer.

For coarse meshes (low levels of refinement), the computed boundary layer is primarily due to numerical viscosity, the corresponding viscous quantities, especially near the leading edge, being far from those obtained in the previous example. Figure 7.27 presents, for instance, the profile of heat flux along the solid wall (compare with Fig 7.23).

The situation drastically changes, however, when a layer of second, third, and fourth order elements are introduced into the mesh (see Fig. 7.28). The corresponding heat flux profile, shown in Fig. 7.29, agrees qualitatively with the results obtained in the previous example.

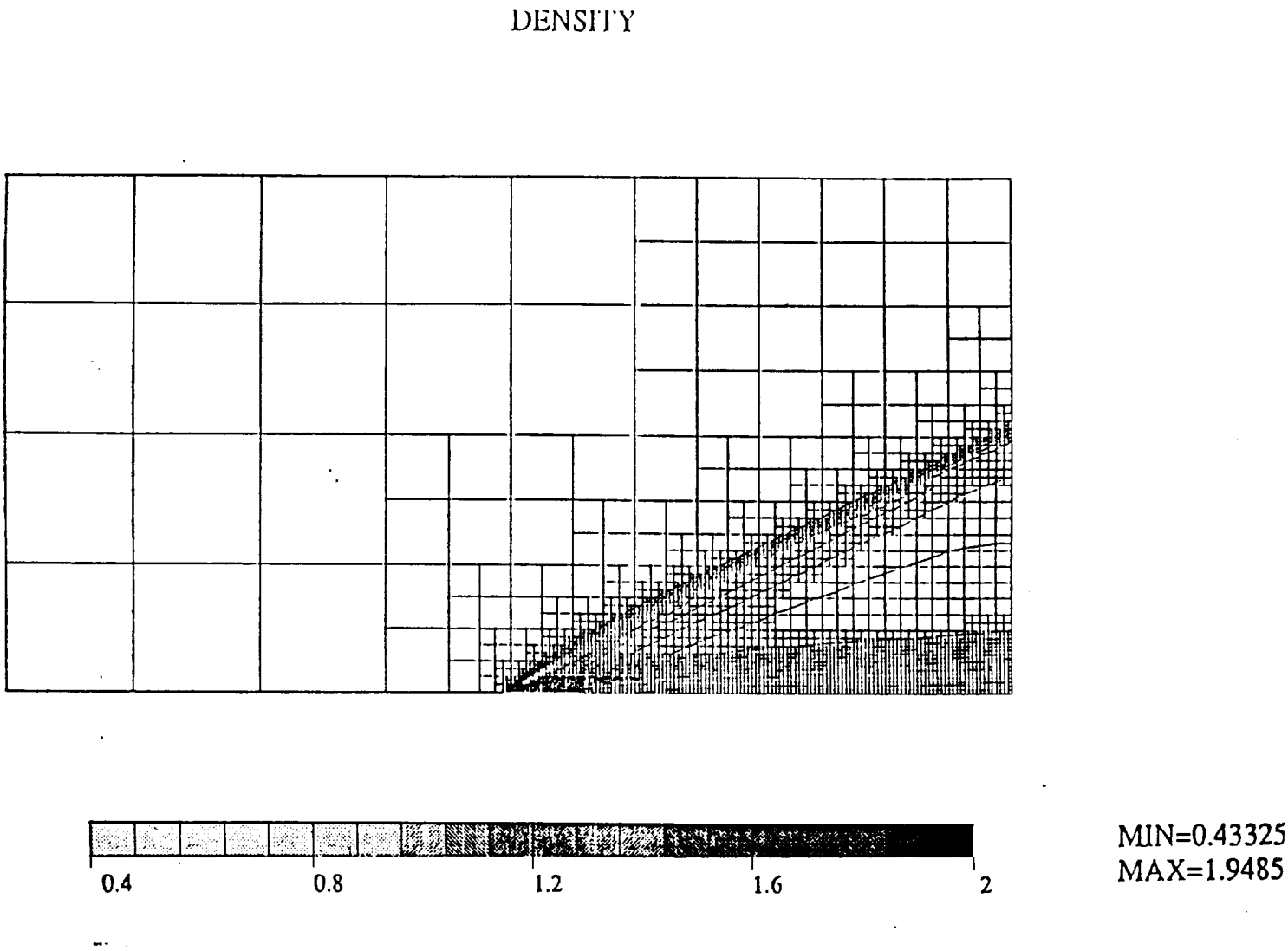


Figure 7.25: Flat plate problem with $Re = 10,000$. Density contours and optimal mesh of linear elements with the maximum level of element = 7.

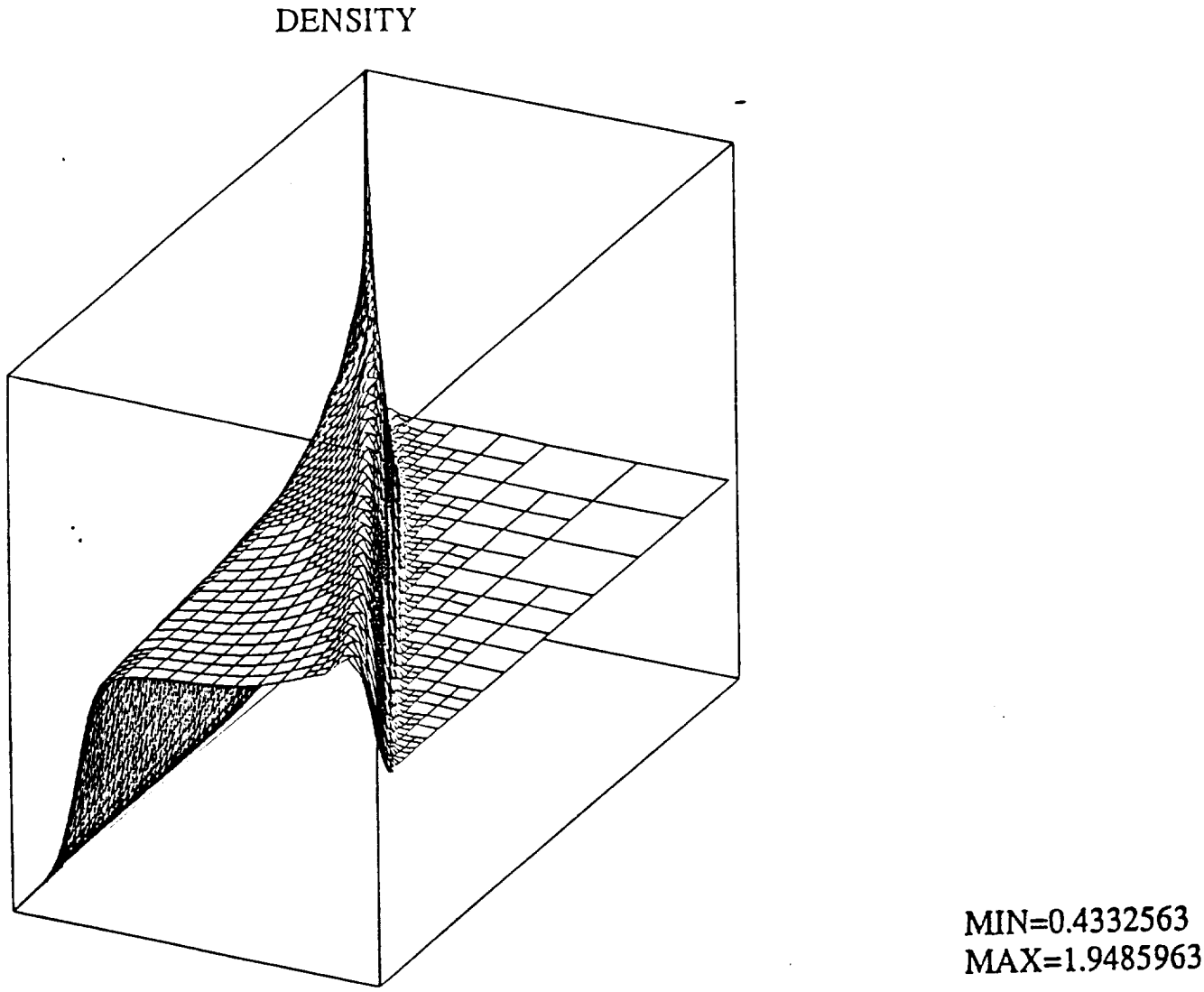


Figure 7.26: Flat plate problem with $Re = 10,000$. A three–dimensional perspective of the density function on the optimal mesh of linear elements.

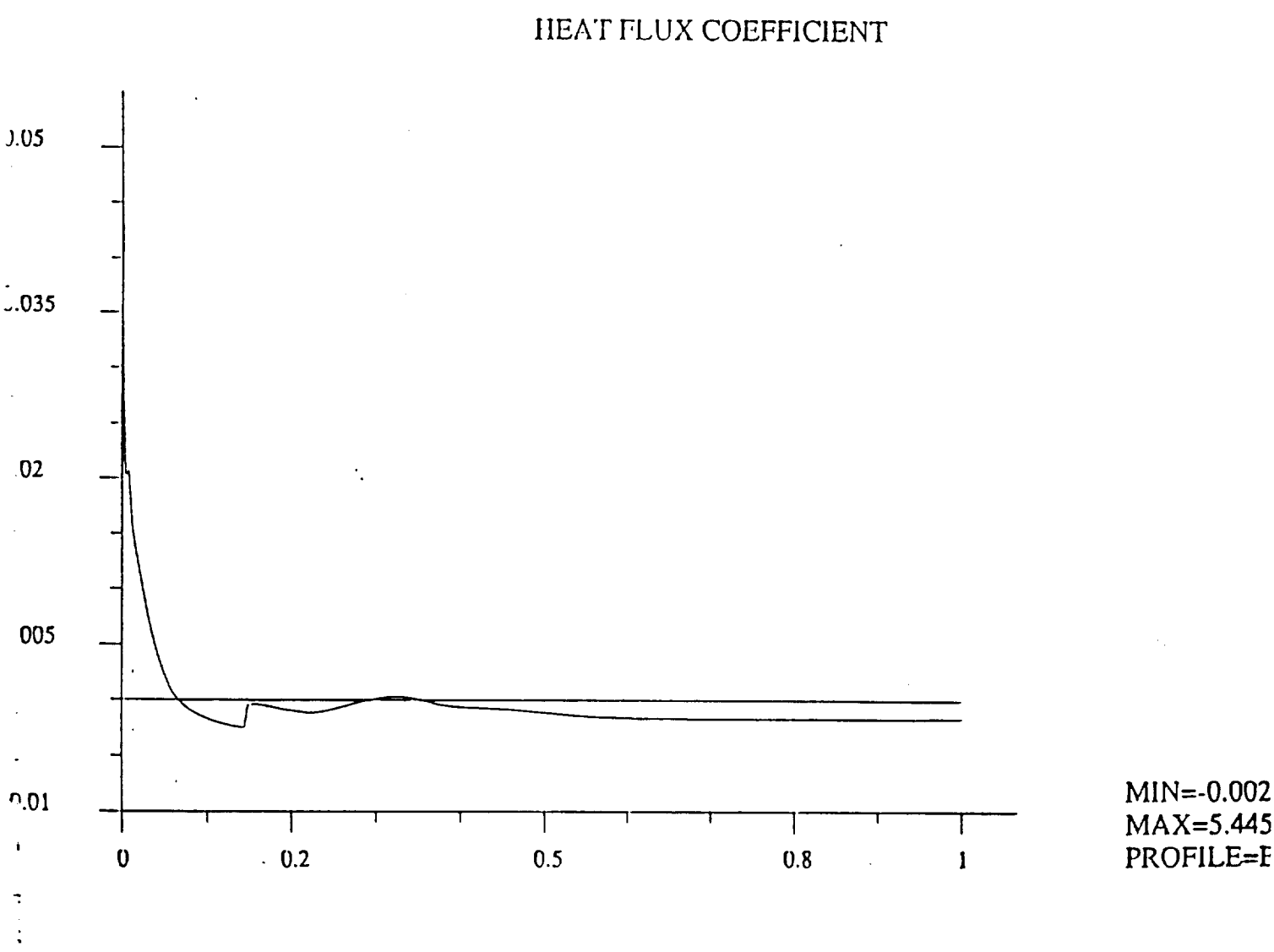


Figure 7.27: Flat plate problem with $Re = 10,000$. Heat flux coefficient profile along the plate for the mesh of linear elements.

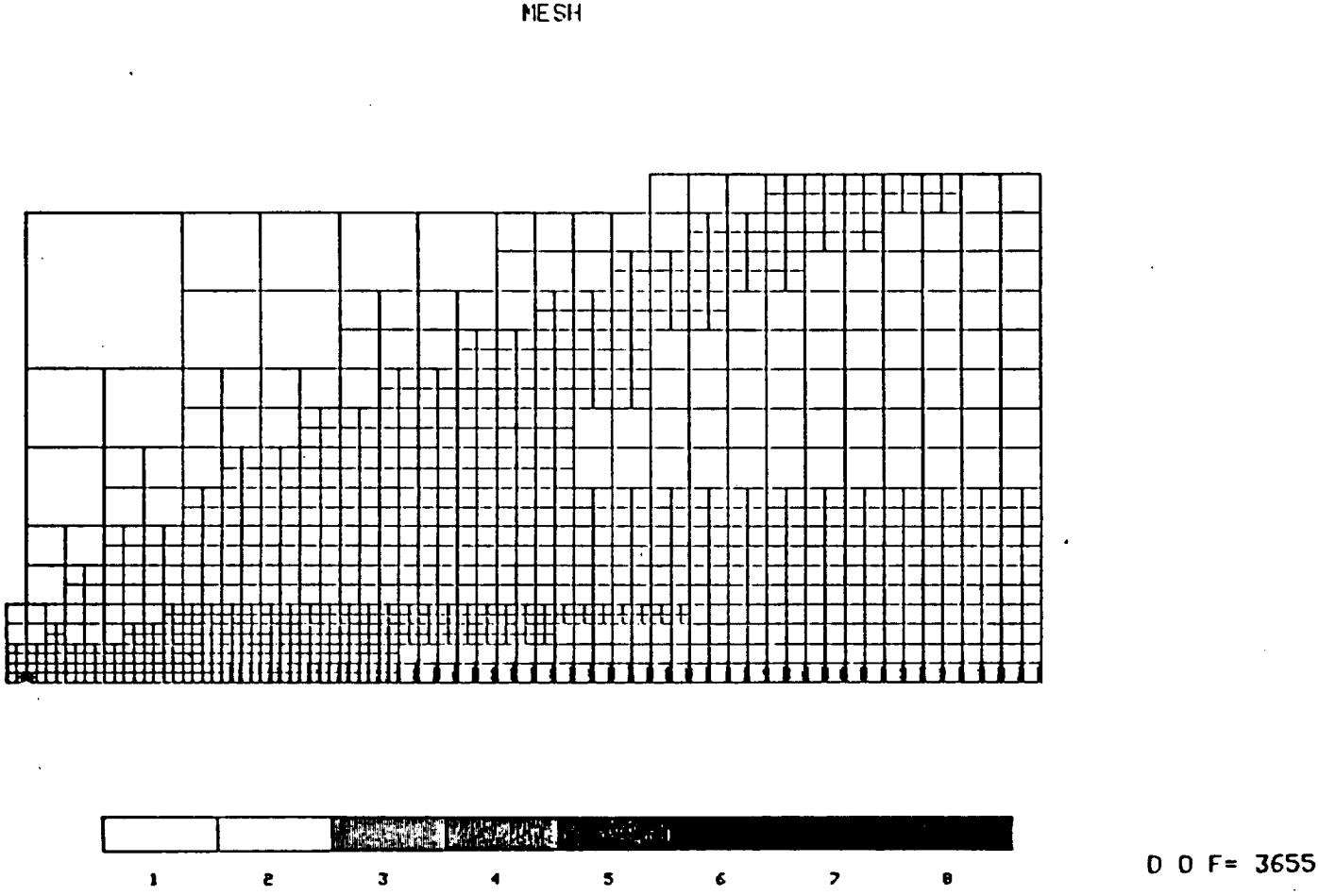


Figure 7.28: Flat plate problem with $Re = 10,000$. Blowup of the optimal mesh enriched in the boundary layer.

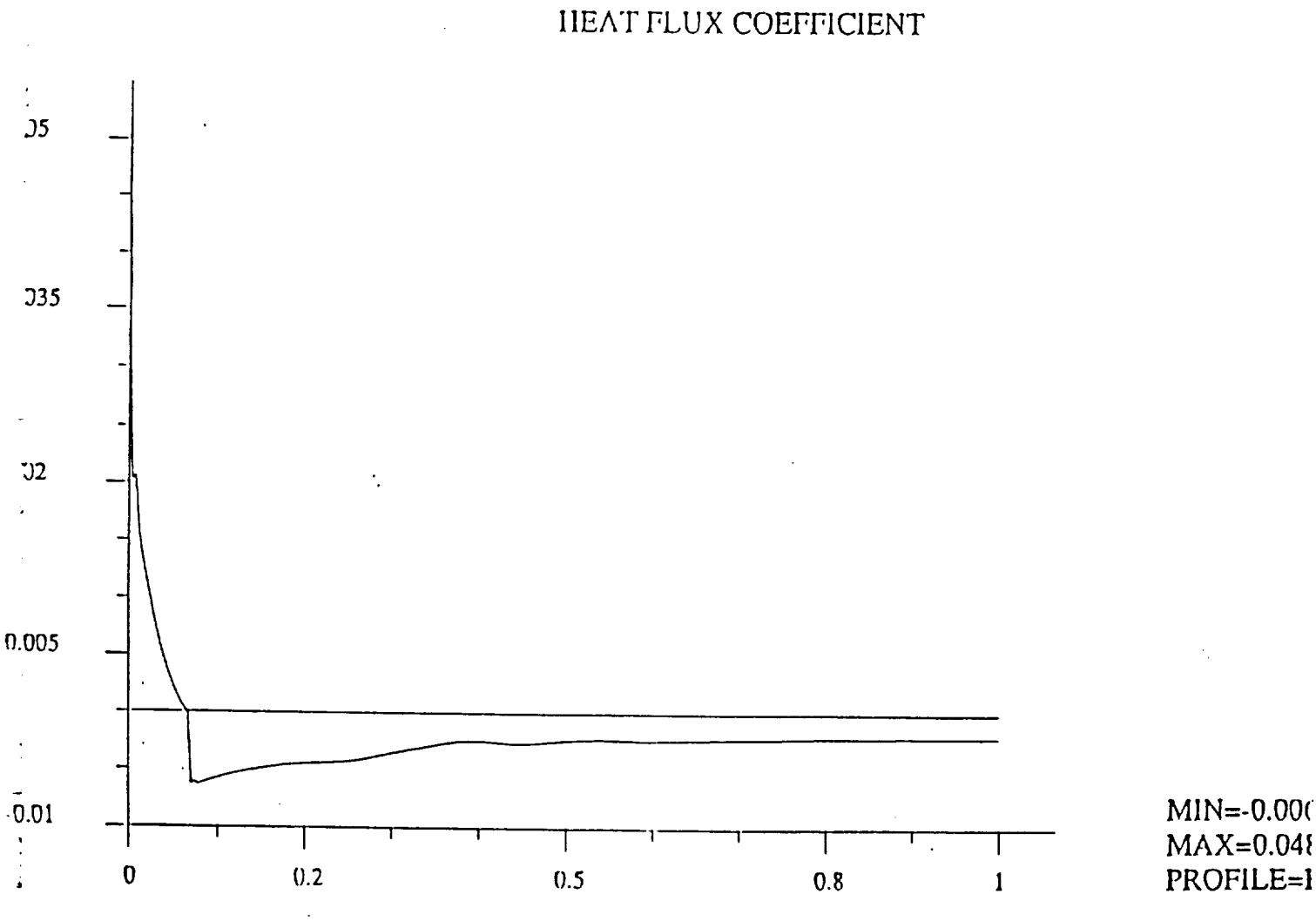


Figure 7.29: Flat plate problem with $Re = 10,000$. Heat flux coefficient profile along the plate for the enriched mesh.

Example 5: The Holden Compression Corner Problem, $M = 5$, $Re = 30,000$

The next two–dimensional test case modeled was the viscous flow over a compression corner. The problem was solved for the following data:

$$\begin{aligned} M &= 5 \\ Re &= 30,000 \\ T_{\text{wall}} &= 288°K \\ T_\infty &= 80°K \end{aligned}$$

angle of inclination 15°

The initial mesh consisted of 7×17 linear elements. Three levels of h-refinements were subsequently performed leading to a good resolution of the shocks inside the computational domain. Finally, layers of quadratic, cubic, and fourth order anisotropic elements were introduced along the solid wall boundary. The elements in the neighborhood of the stagnation point were not enriched as we experienced some stability problems with higher order elements in this area. The final h-p mesh is shown in Fig. 7.30.

The solution, contours of density and profiles of the density, velocity, temperature and pressure coefficient are shown in Figs. 7.31–7.43. As in the previous example a stabilizing effect of higher order approximations on the behavior or pressure can be observed. The profiles of the heat flux and the skin friction are shown in Figs. 7.44 and 7.45. A dramatic change in the quality of these two fluxes is observed as we move from the section of the boundary with linear elements only to the enriched part.

The higher order approximation resulted also in a satisfactory resolution of another fine feature of the solution: the recirculation of the flow. This is shown in Figs. 7.46 and 7.47 where the directions of the flow and the negative contours of u_1-velocity in the zone of recirculation are presented.

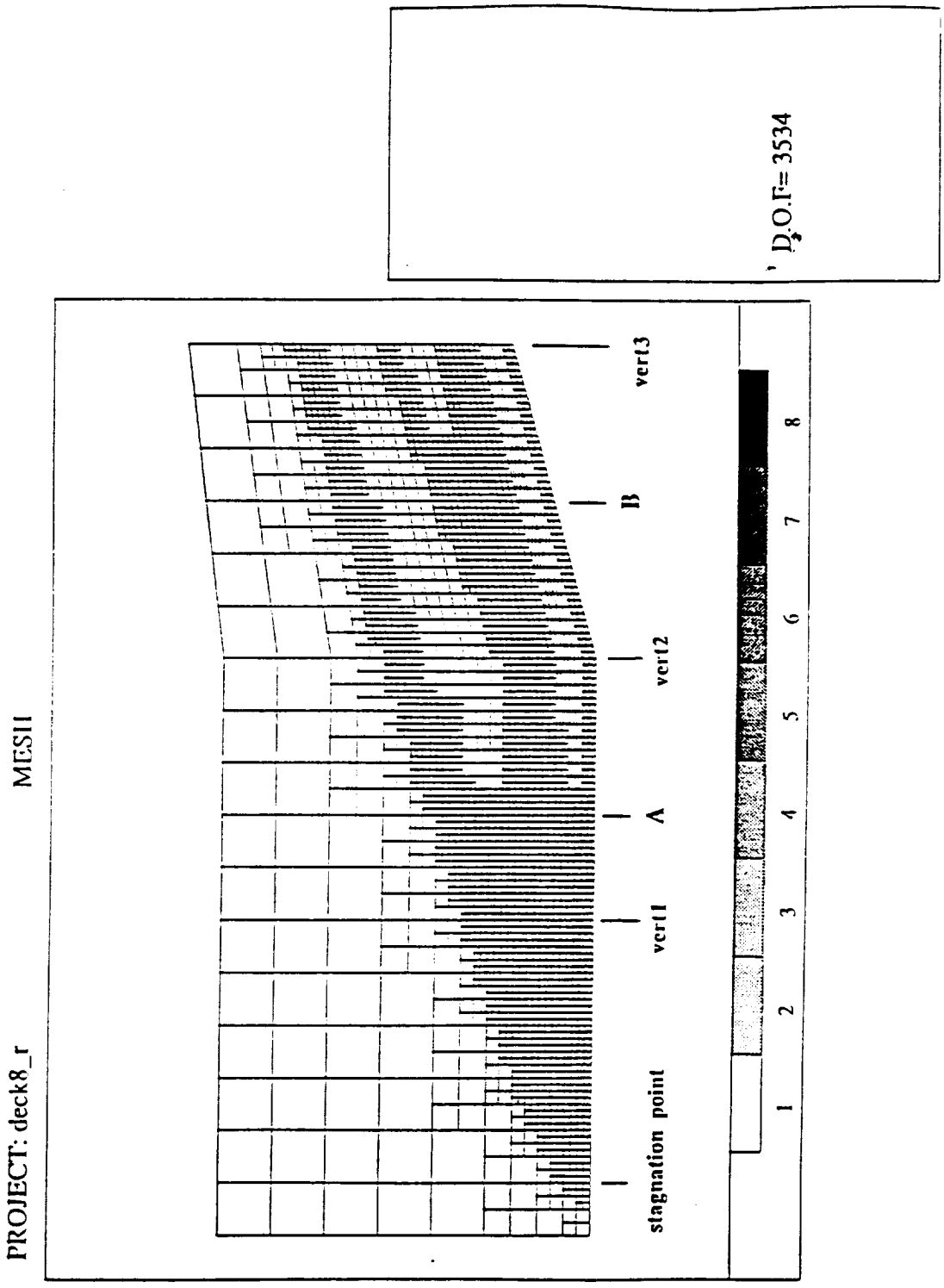


Figure 7.30: Holden's compression corner problem. *h-p* adaptive mesh.

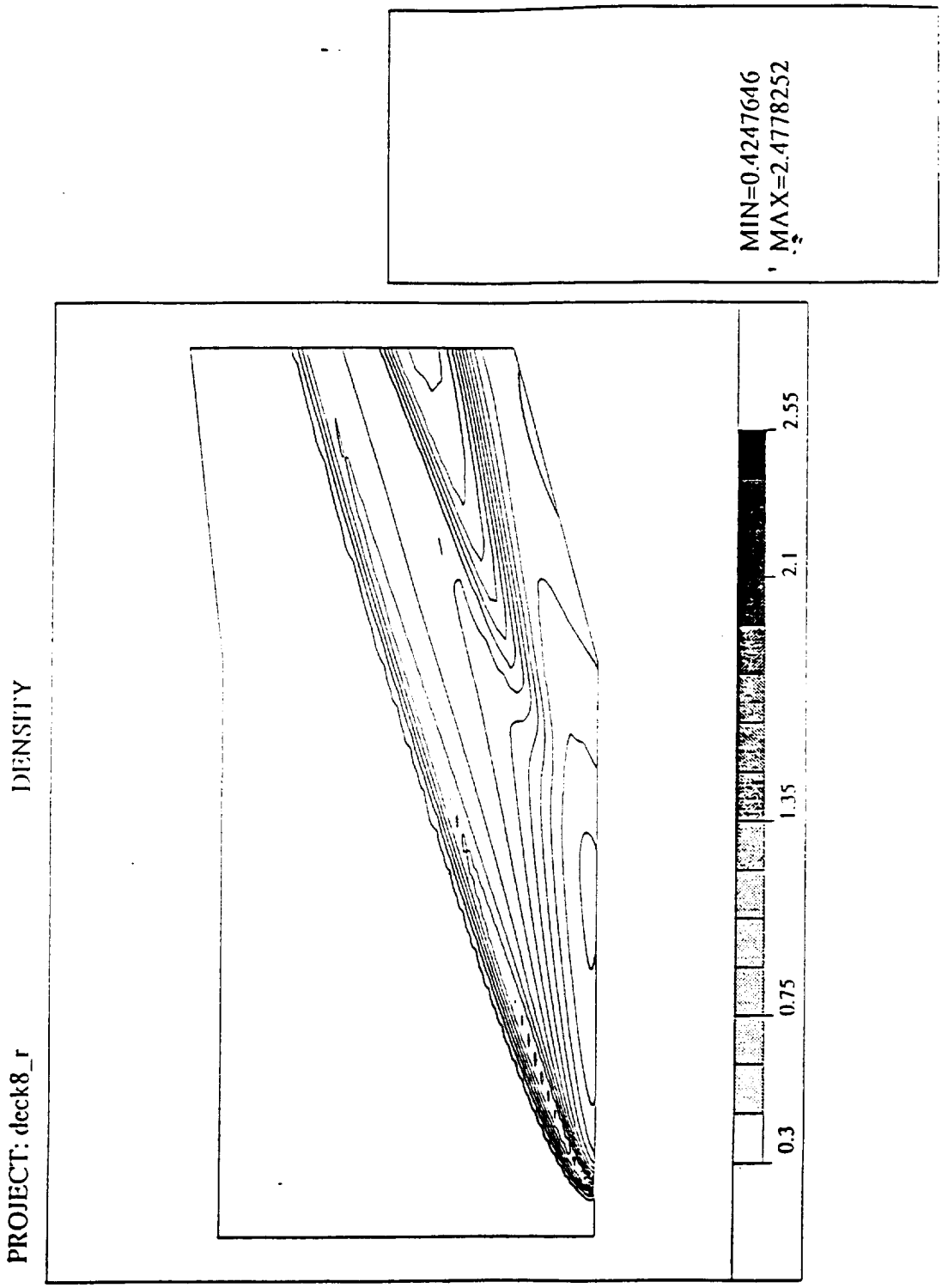


Figure 7.31: Holden's compression corner problem. Density contours.

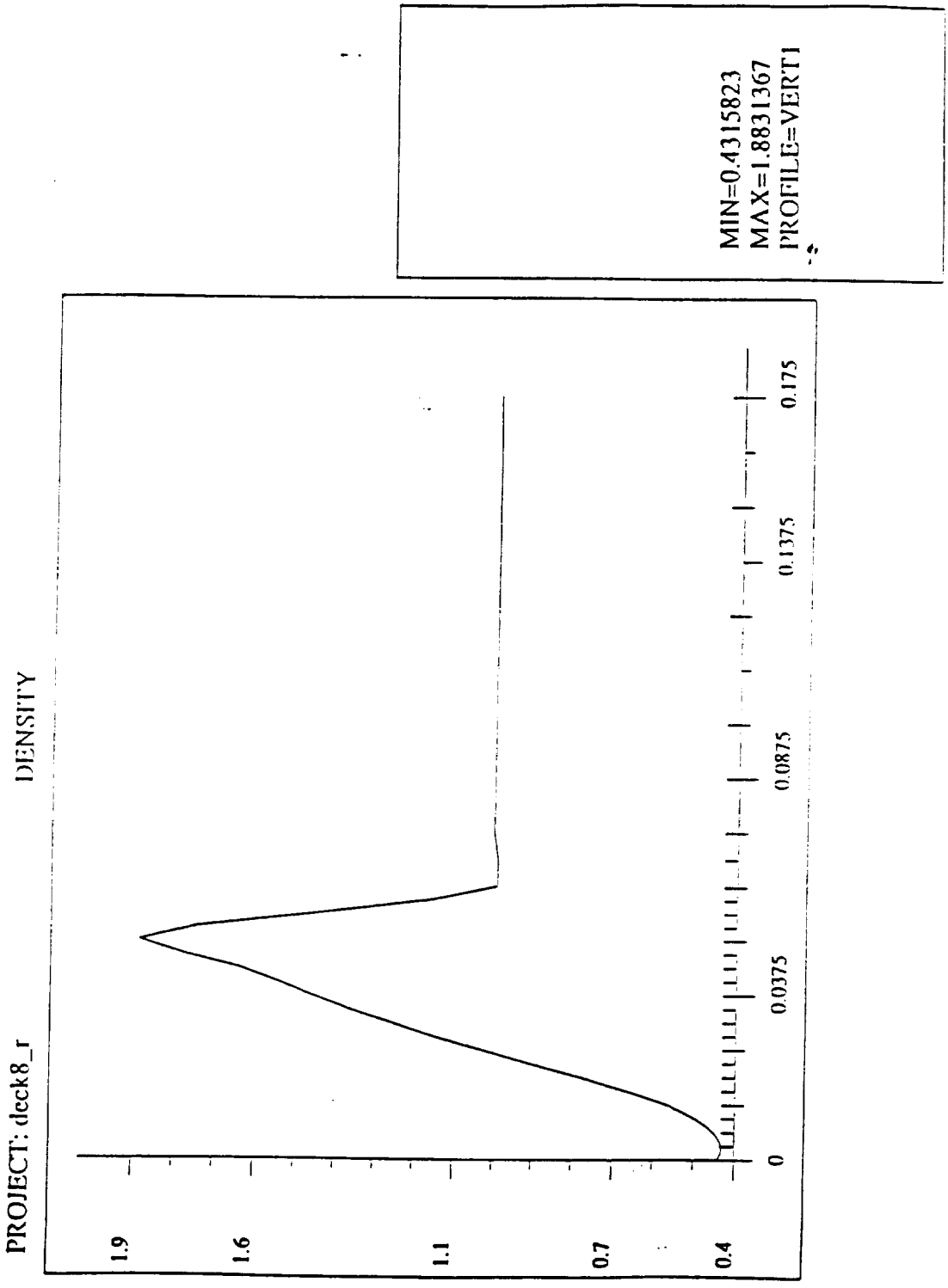


Figure 7.32: Holden's compression corner problem. Profile of density along vert 1.

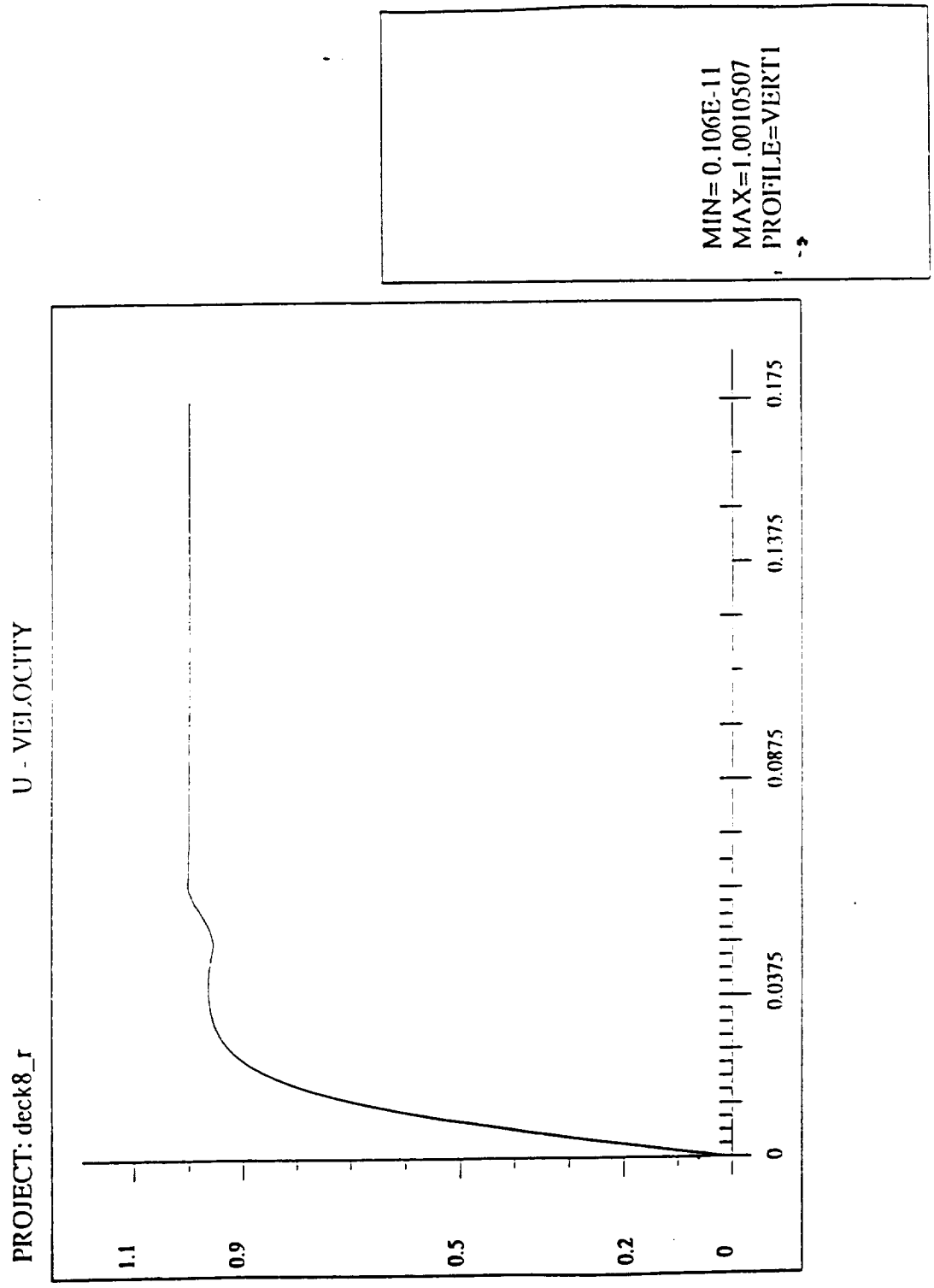


Figure 7.33: Holden's compression corner problem. Profile of velocity along vert 1.

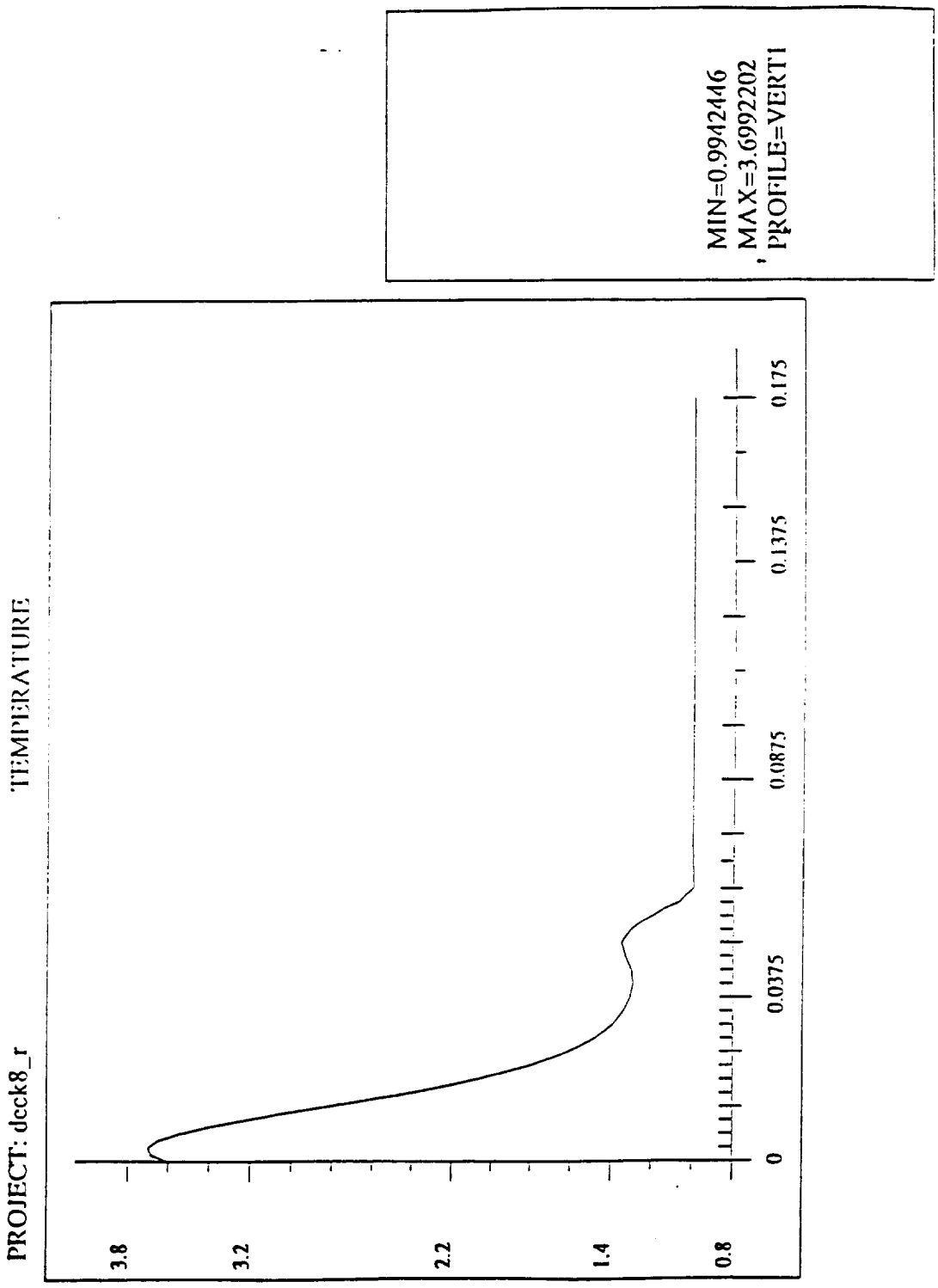


Figure 7.34: Holden's compression corner problem. Profile of temperature along vert 1.

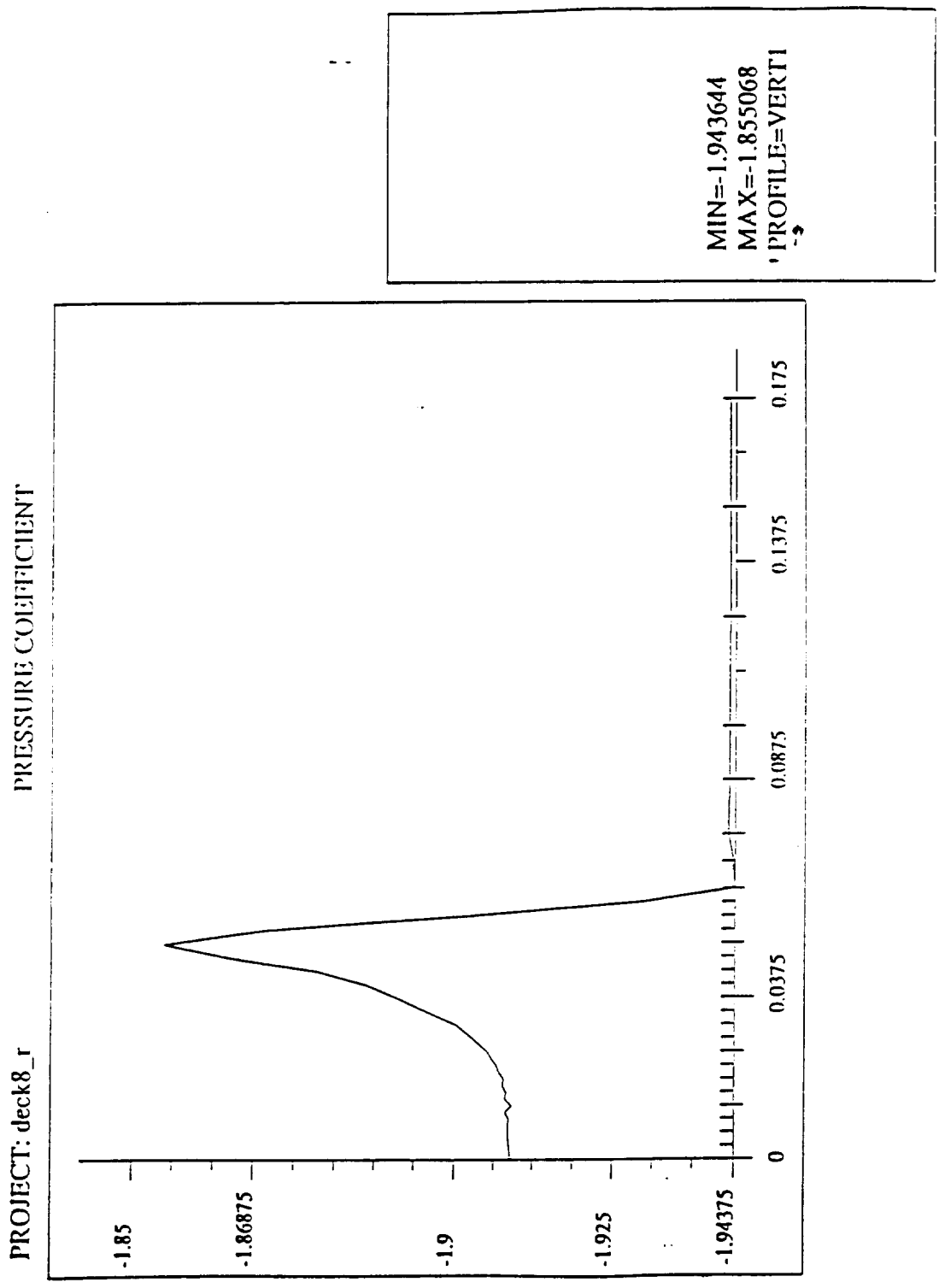


Figure 7.35: Holden's compression corner problem. Profile of pressure coefficient along vert 1.

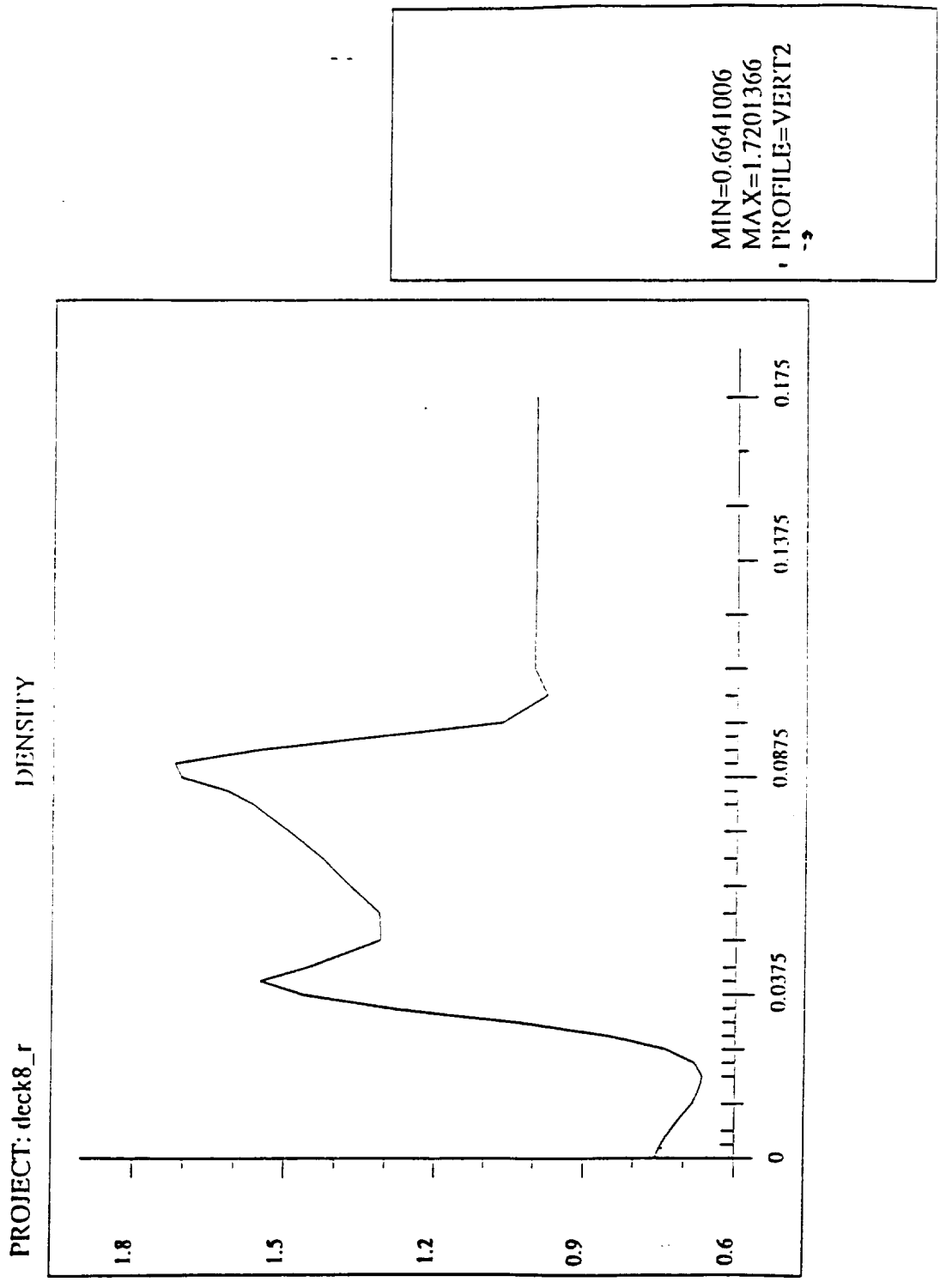


Figure 7.36: Holden's compression corner problem. Profile of density along vert 2.

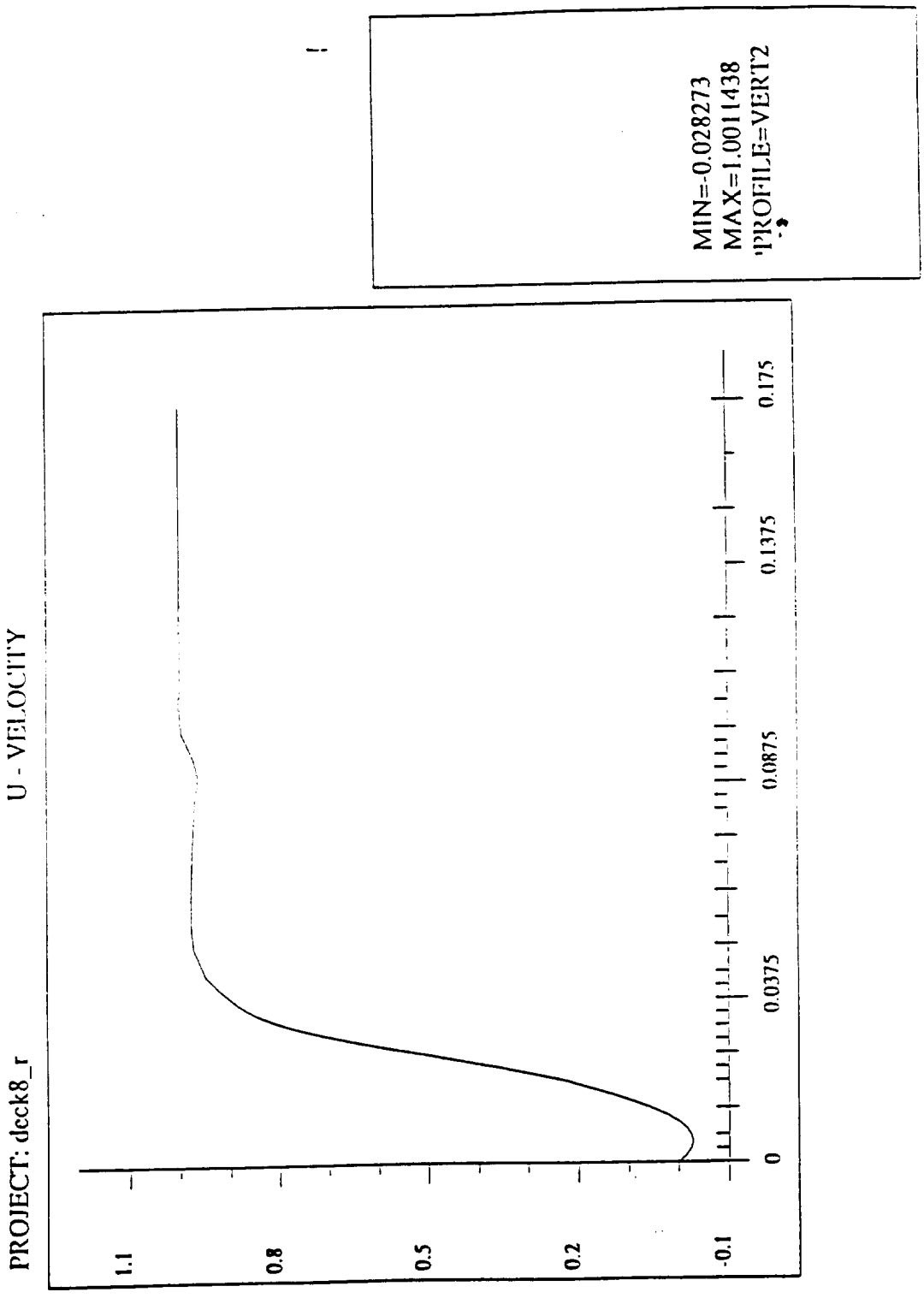


Figure 7.37: Holden's compression corner problem. Profile of velocity along vert 2.

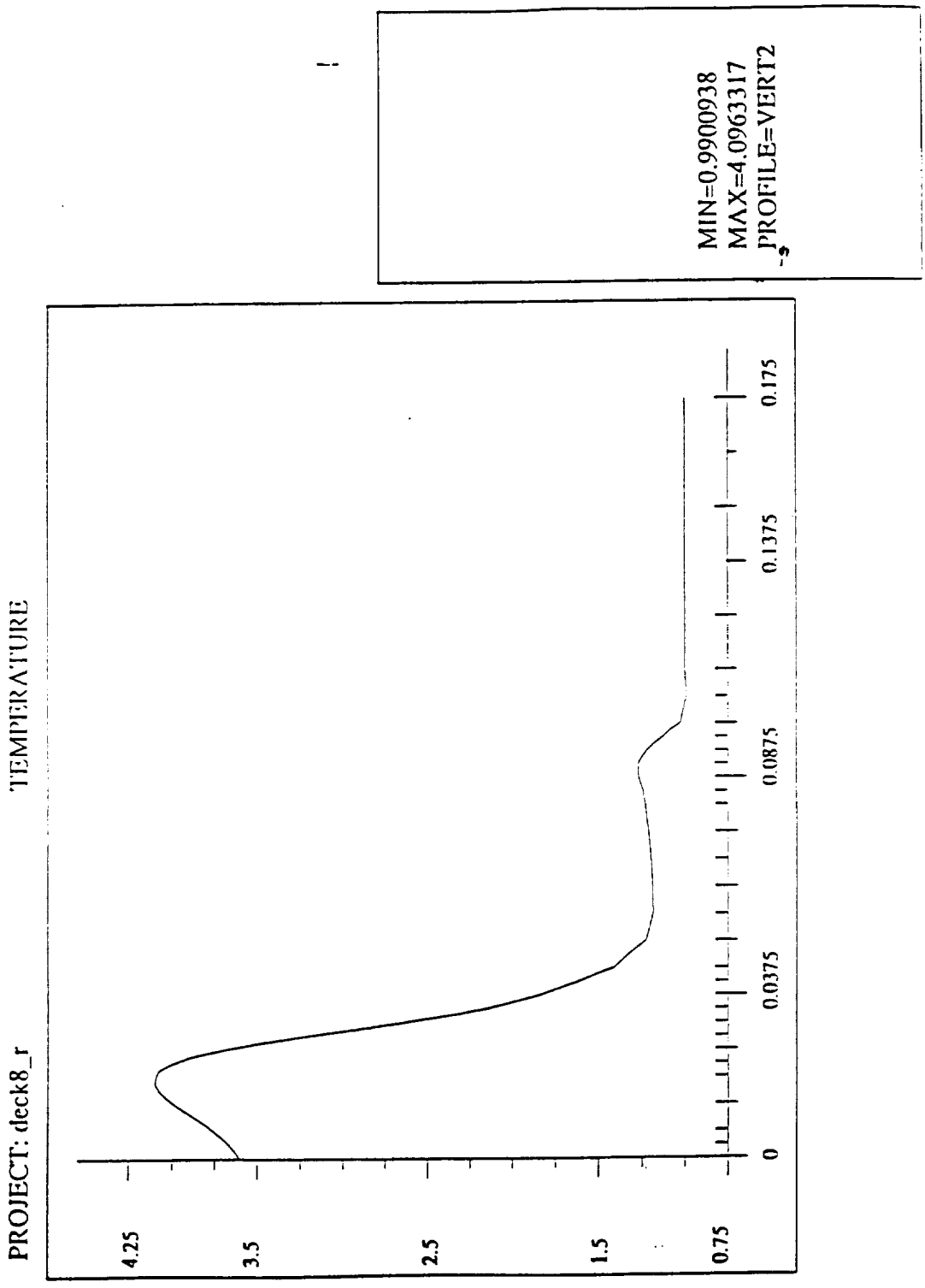


Figure 7.38: Holden's compression corner problem. Profile of temperature along vert 2.

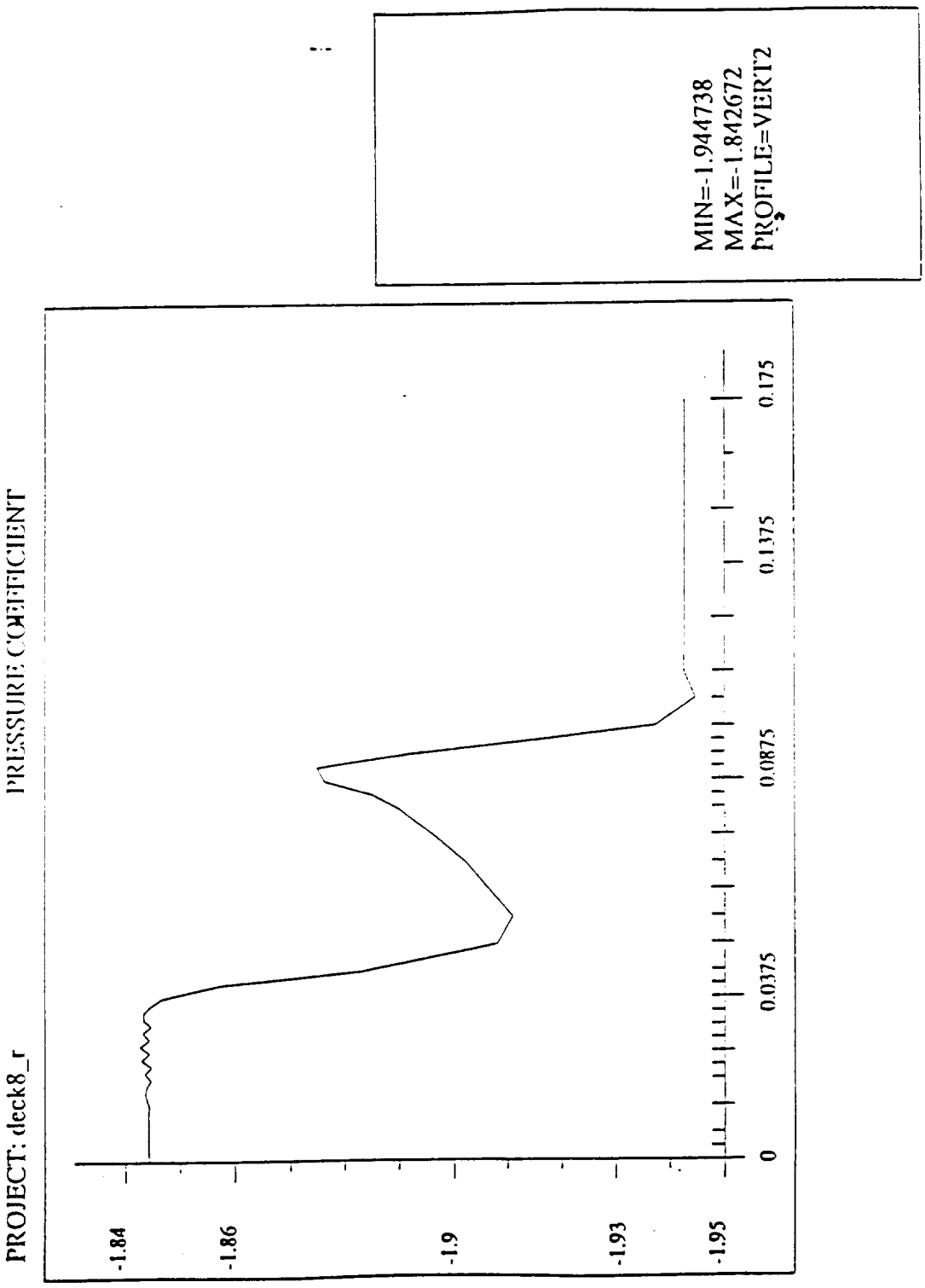


Figure 7.39: Holden's compression corner problem. Profile of pressure coefficient along vert 2.

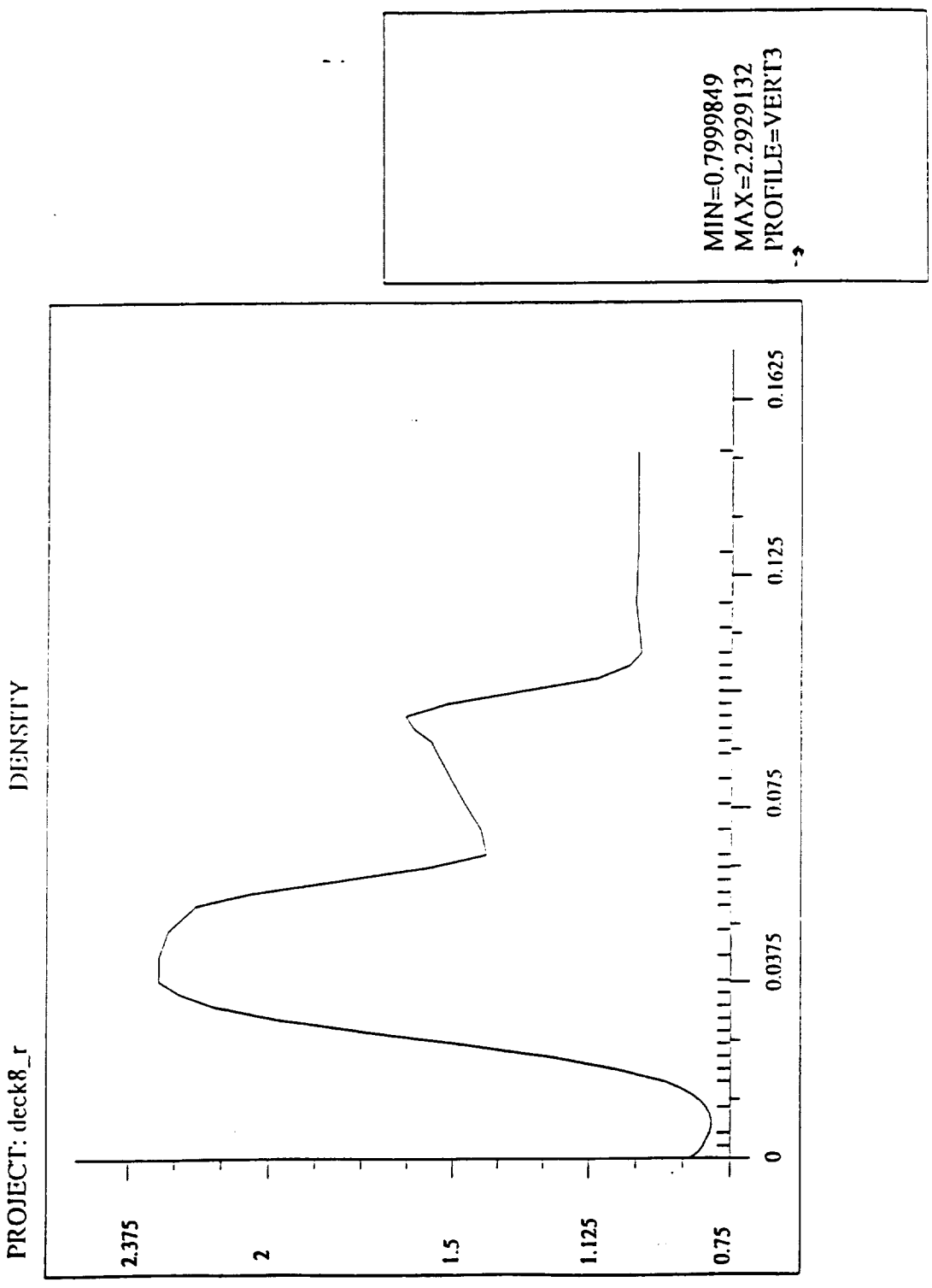


Figure 7.40: Holden's compression corner problem. Profile of density along vert 3.

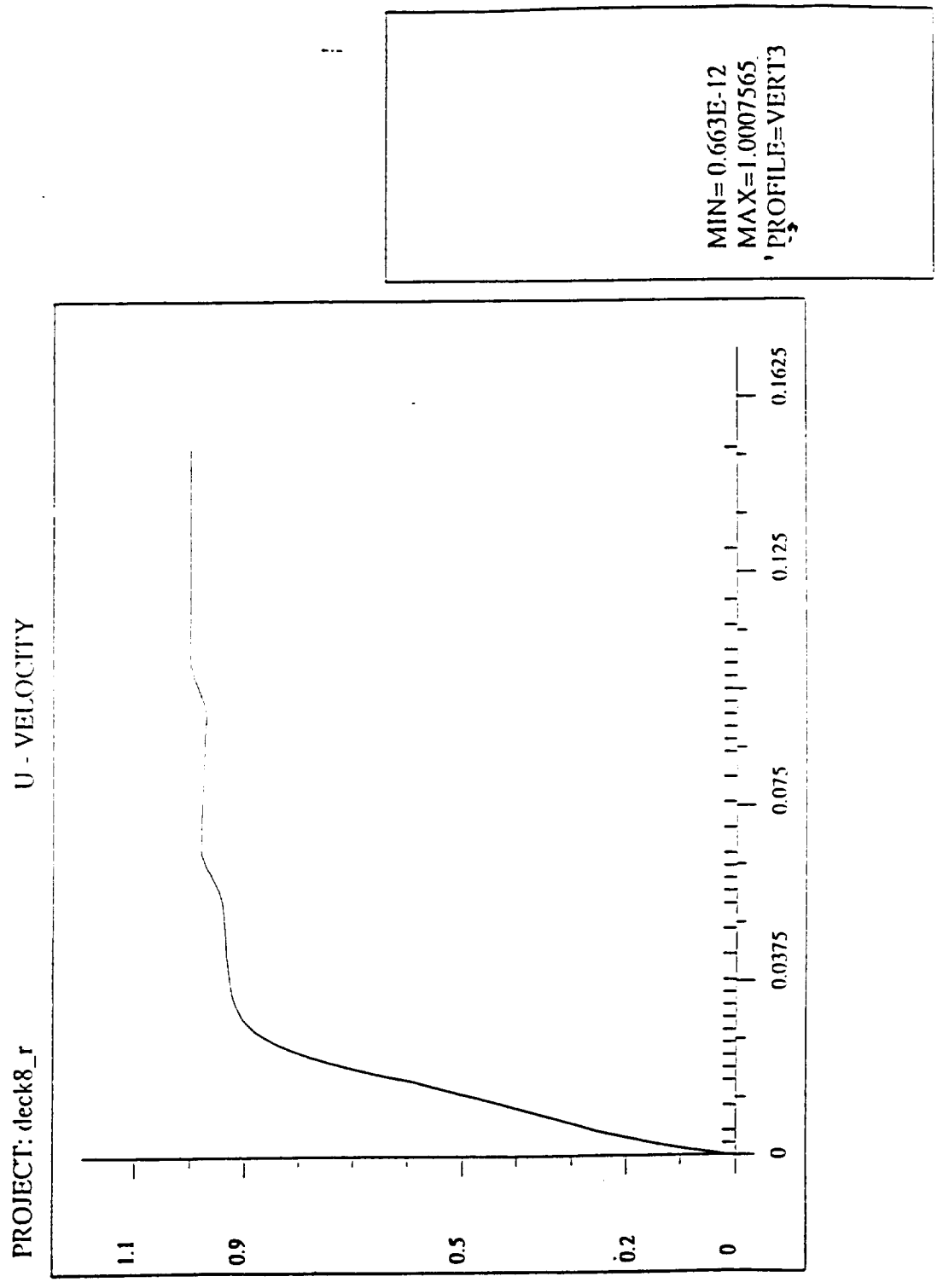


Figure 7.41: Holden's compression corner problem. Profile of velocity along vert 3.

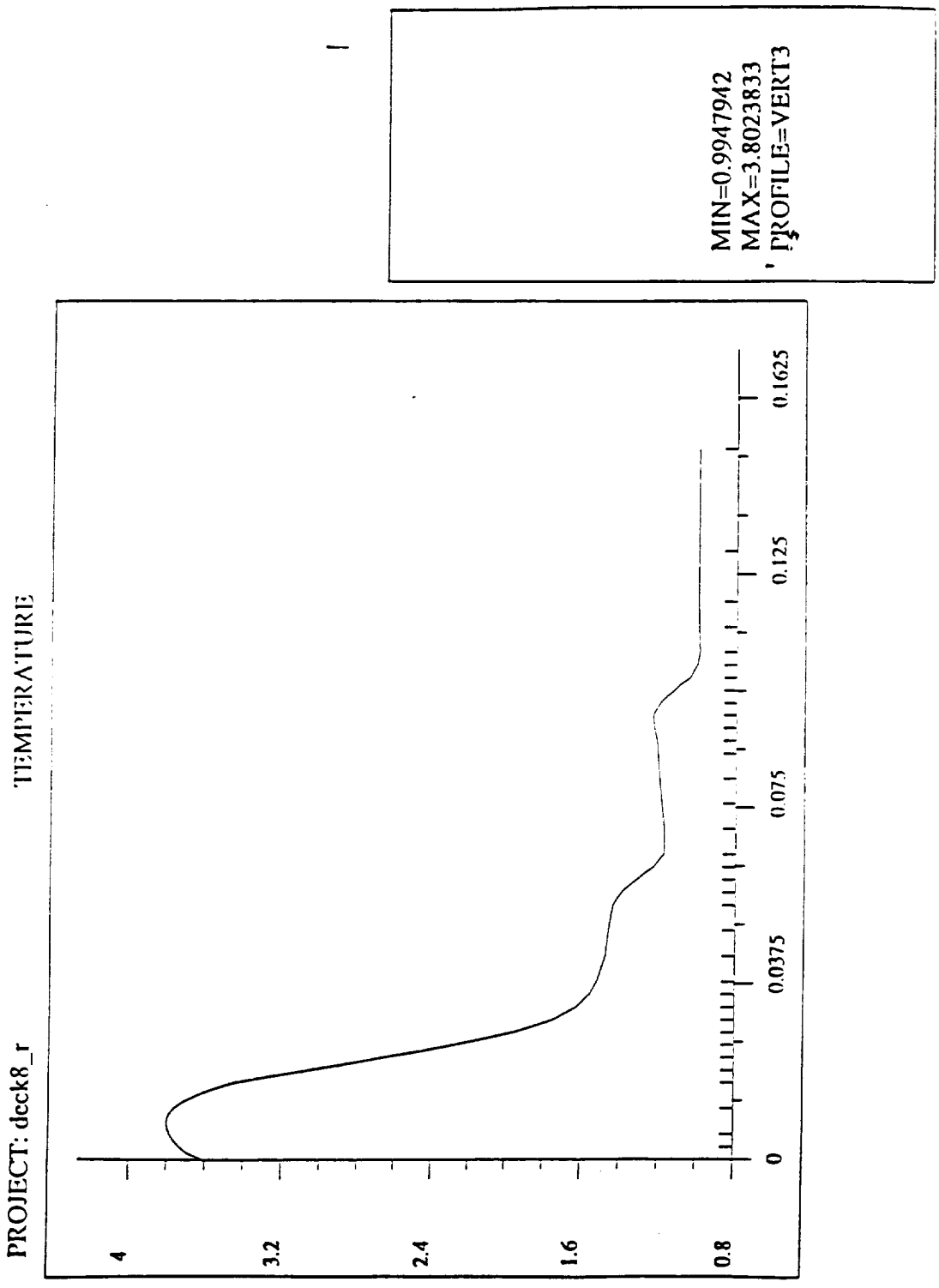


Figure 7.42: Holden's compression corner problem. Profile of temperature along vert 3.

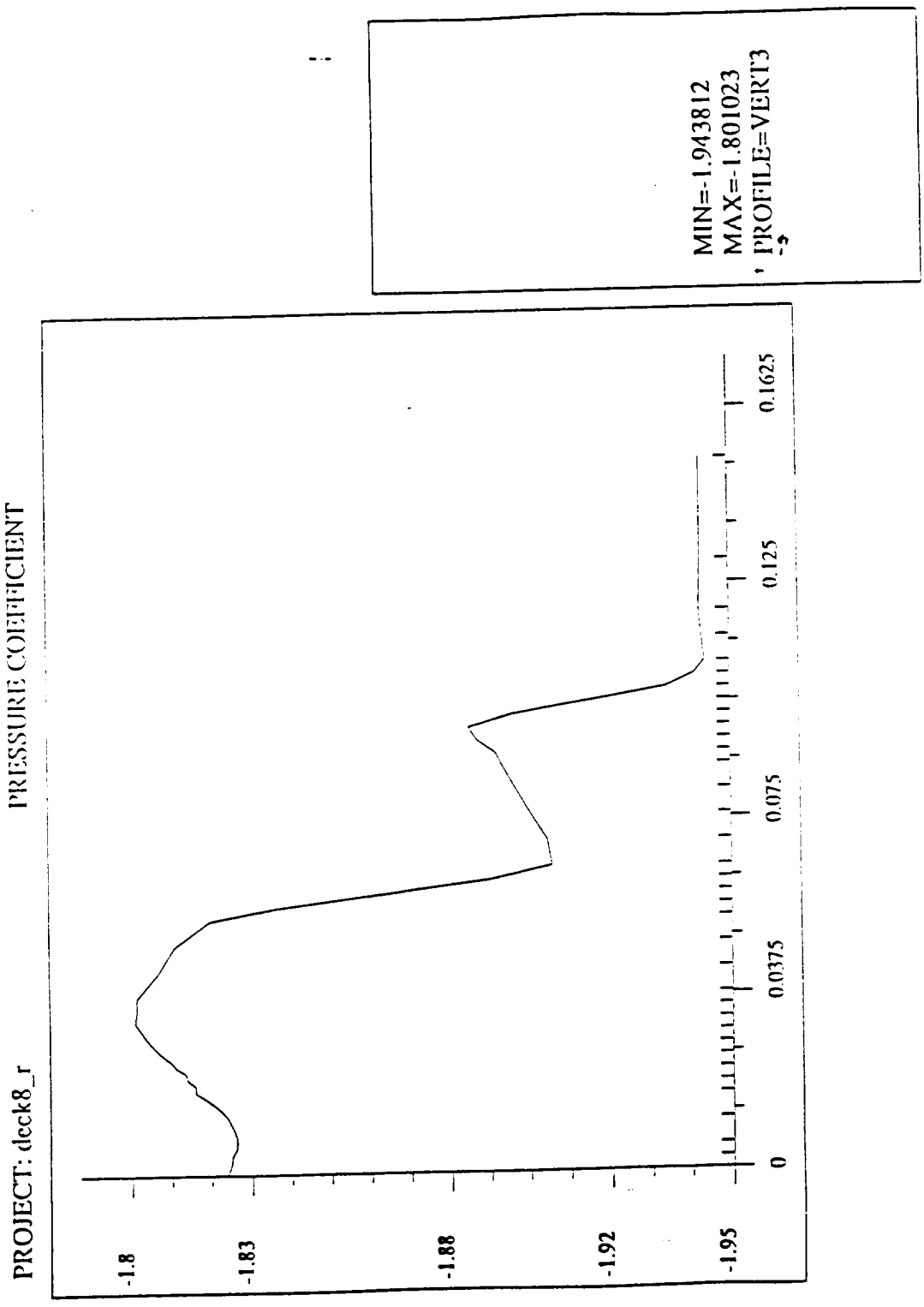


Figure 7.43: Holden's compression corner problem. Profile of pressure coefficient along vert 3.

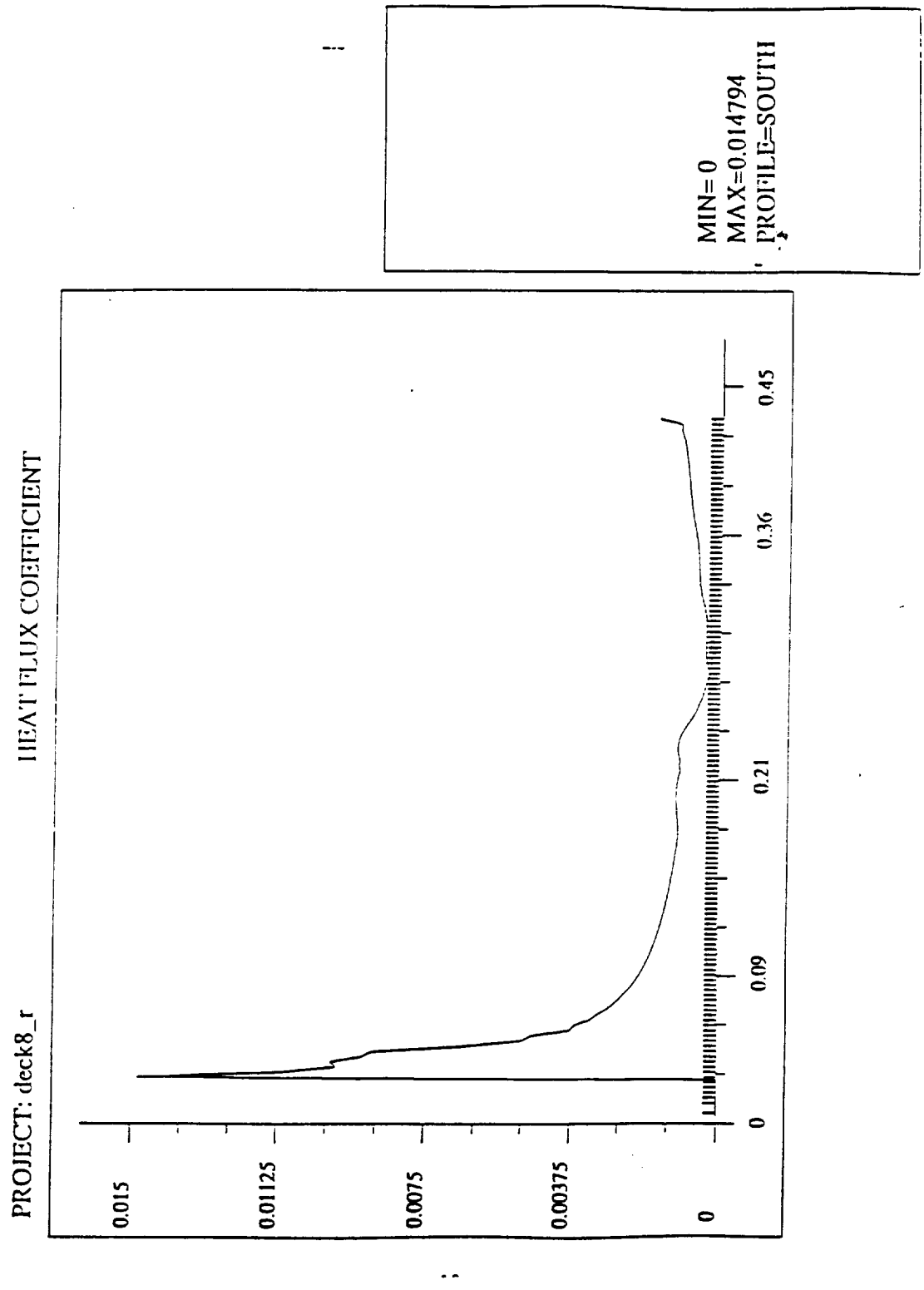


Figure 7.44: Holden's compression corner problem. Profile of heat flux coefficient along the plate.

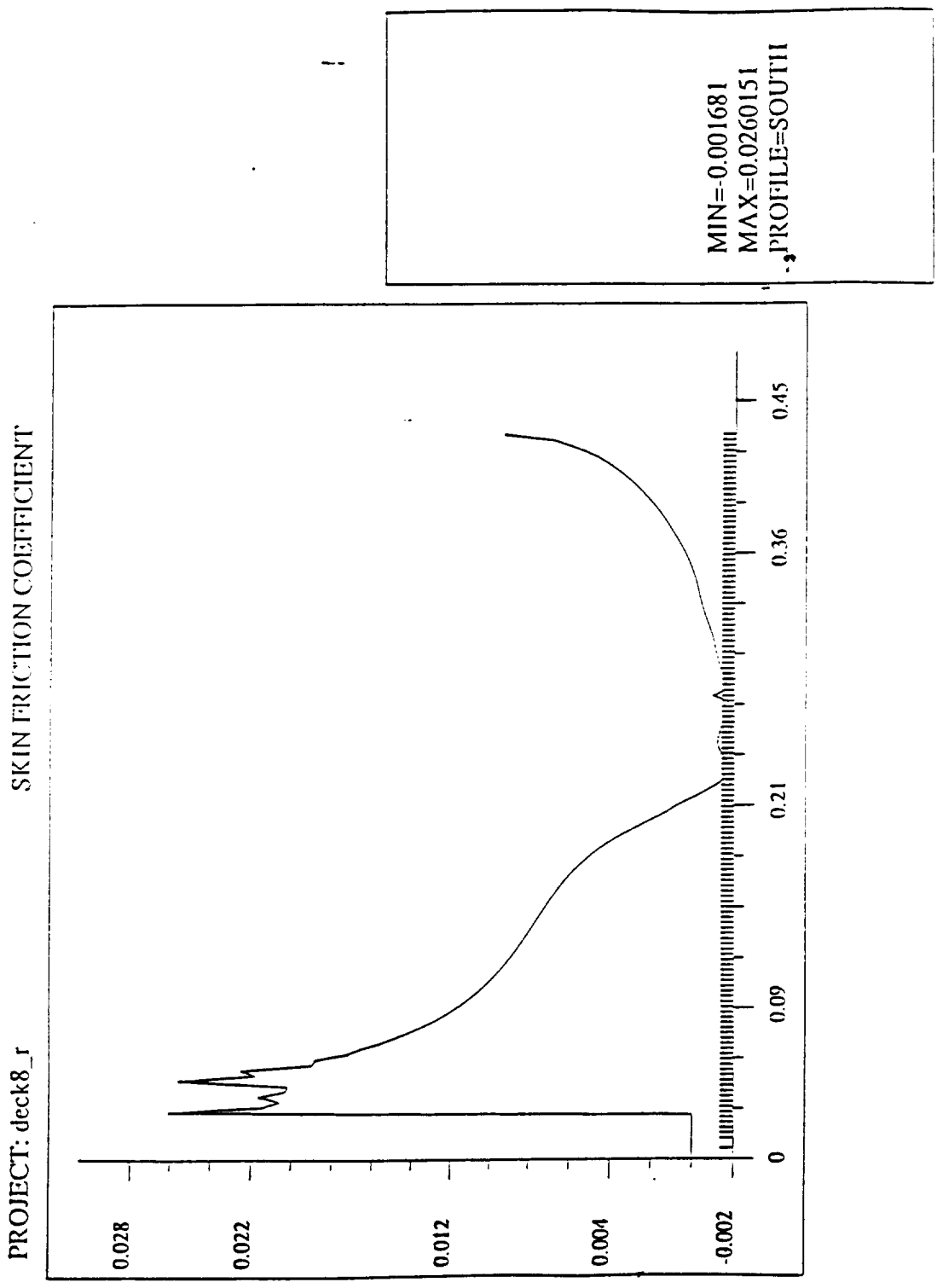


Figure 7.45: Holden's compression corner problem. Profile of skin friction coefficient along the plate.

C-3

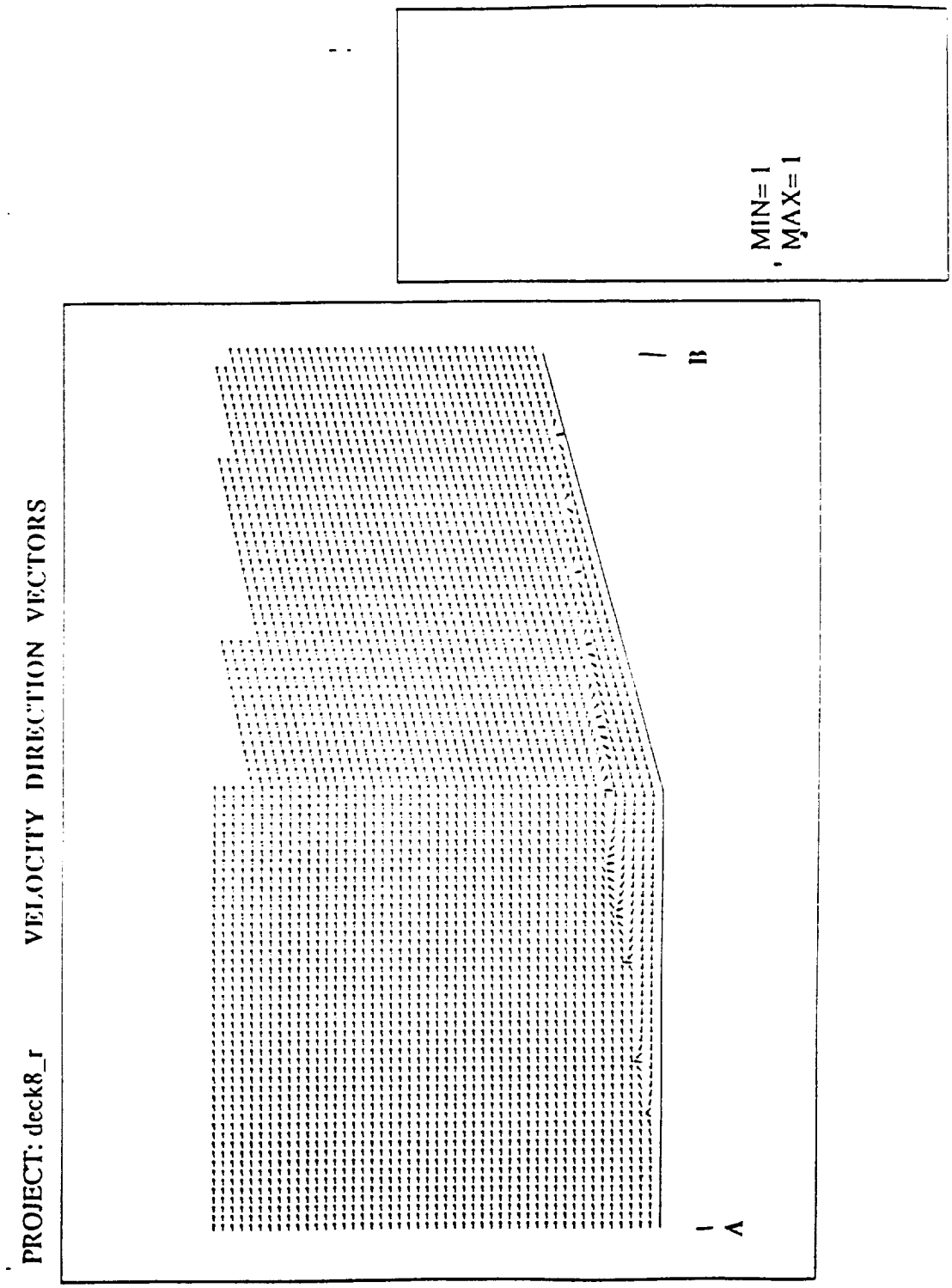


Figure 7.46: Holden's compression corner problem. Recirculation: velocity vectors near the corner.

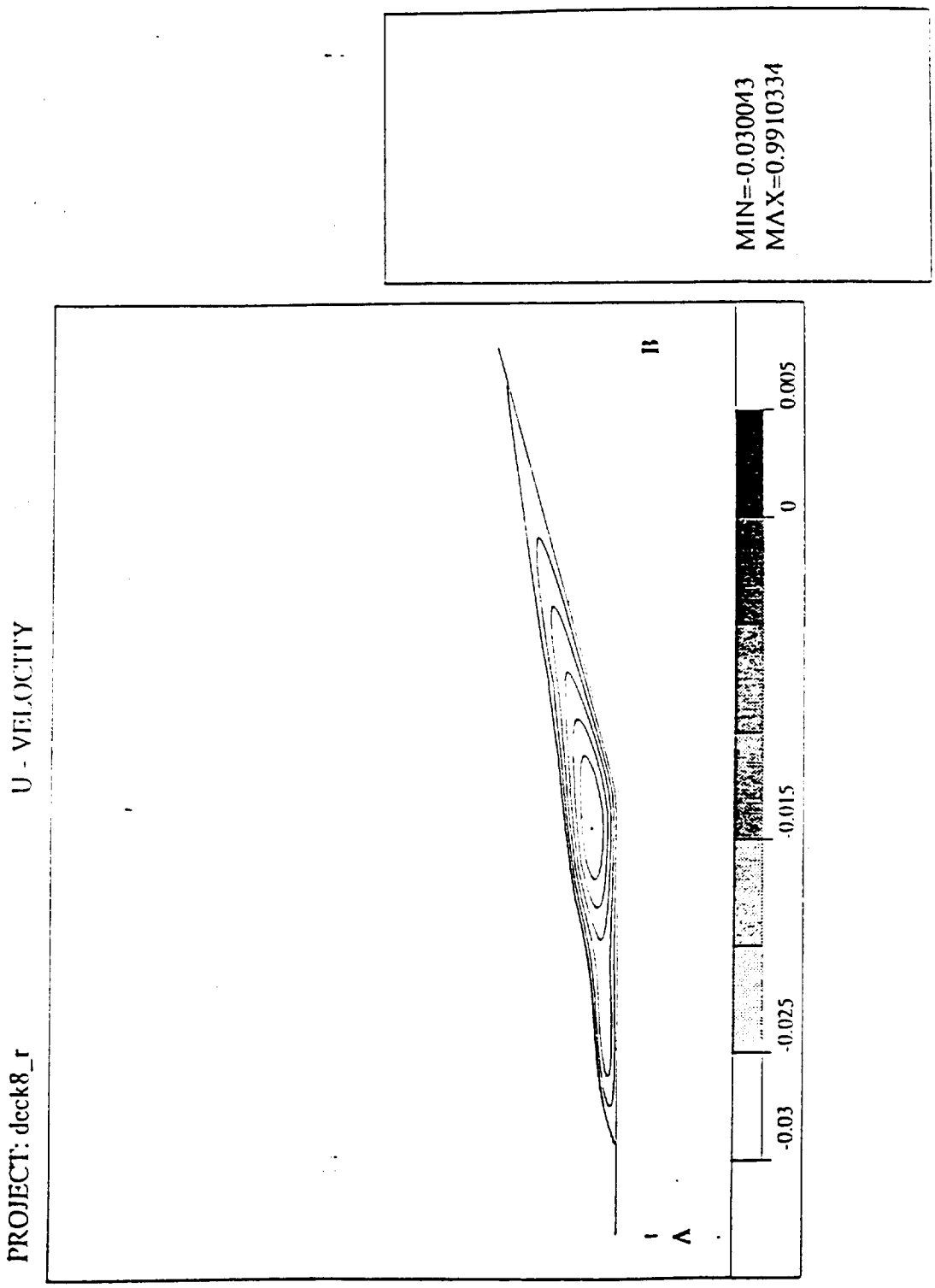


Figure 7.47: Holden's compression corner problem. Recirculation: contours of the u_1 component of the velocity.

Example 6: Inviscid Flow Past a 20° Wedge, $M = 3$

Our first three–dimensional test case was the supersonic inviscid flow ($M = 3$) over a two–dimensional wedge (analyzed using the three–dimensional code). The initial mesh consisted of one layer of 4×8 linear elements. On the side surfaces we imposed symmetry boundary conditions, which enforce (weakly) the condition $\partial U/\partial n = 0$, which was intended to enforce the two–dimensional character of the flow. The solution (contours of density) and h-adapted mesh are shown in Fig. 7.48. One can observe that indeed the solution is essentially independent of the y-coordinate direction. Comparing this result with the two–dimensional case (Fig. 7.2), one observes a good qualitative agreement of the flow features and shock angle.

Example 7: Inviscid Flow Around a Sphere, $M = 6$

Next, the supersonic ($M = 6$) flow over a spherical blunt body was solved. In the discretization, symmetry was enforced so that only one quarter of a half sphere is meshed. The initial mesh is generated by appropriately mapping a regular $7 \times 7 \times 4$ rectangular mesh onto the quarter sphere. Such a mapping, if performed to cover exactly the octant of a sphere, can lead to severe distortions of some elements as it has to transform a rectangular domain into a topologically triangular domain; hence, the octant is not covered exactly, leaving some missing sections on the outflow boundary (see Fig. 7.49). On the lower and right planes of Fig. 7.49, symmetry boundary conditions are imposed. Modified Lapidus viscosity is used as the artificial dissipation mechanism in this problem. The h-adapted mesh and computed density contours are shown in Figs. 7.50 and 7.51. The flow is characterized by a bow surface of the shock surrounding the body.

Example 8: Viscous Flow Past a Flat Plate, $M = 3$

The classical two-dimensional flat plate problem was also modeled using the three-dimensional code. The data for the problem are: $M = 3$, $Re = 500$, $T_\infty = 80K$, $T_{\text{wall}} = 280°K$. The computational domain was discretized initially by one layer of 4×8 elements. After converging to a steady state solution the elements along the plate were refined and enriched to second and third order. Anisotropic p-enrichments are used, introducing higher order approximations only in the direction perpendicular to the wall, so as to significantly improve the approximation of the boundary layer. Locally, the structure of the boundary layer is close to that of the one–dimensional case, the largest gradients being in a direction perpendicular to the wall. The enriched mesh is shown in Fig. 7.52, where anisotropically enriched elements are marked by shading their edges in the direction of enrichment.

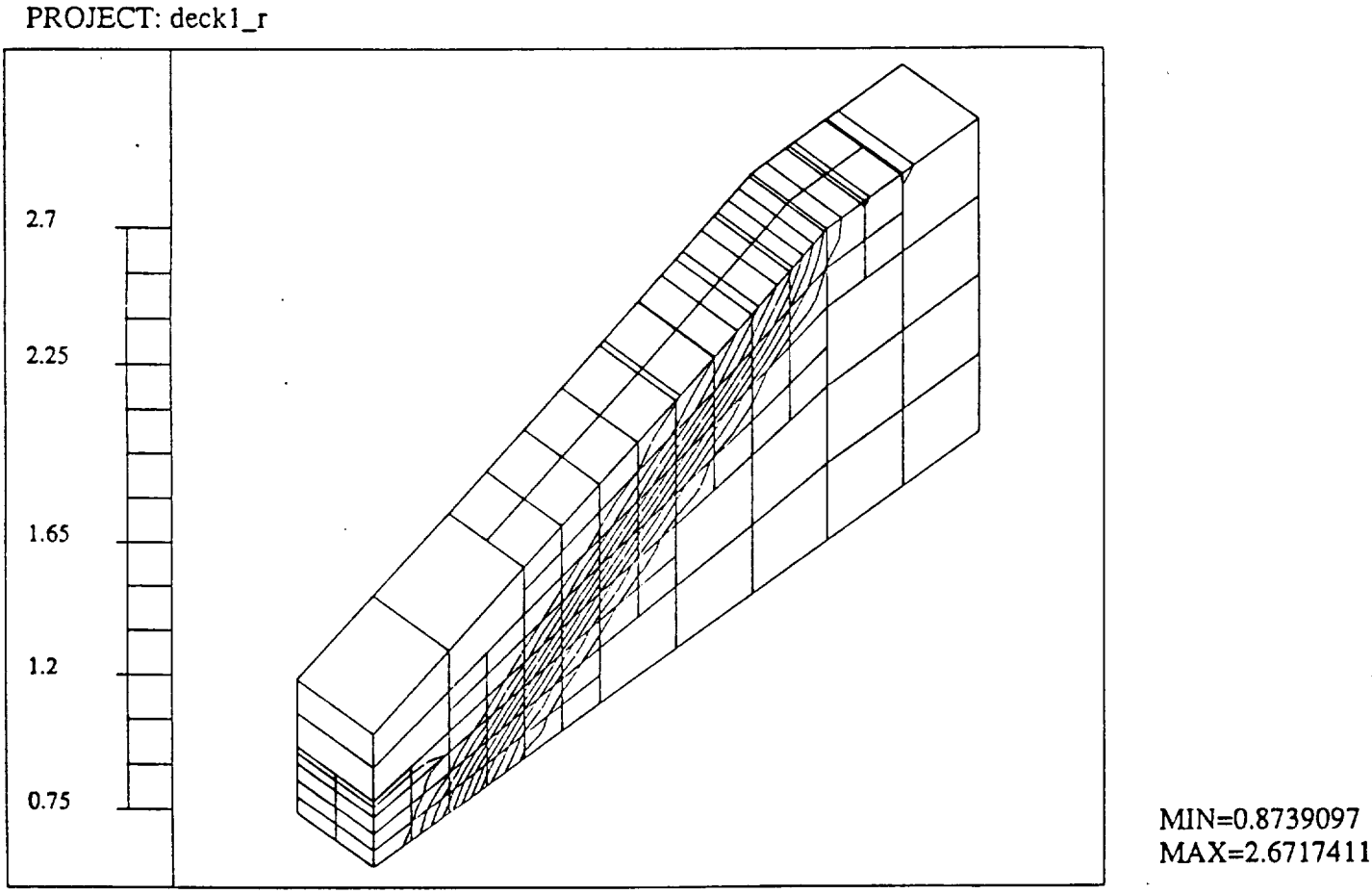


Figure 7.48: Inviscid flow past a 20° wedge, $M = 3$. Density contours and an h-adapted mesh.

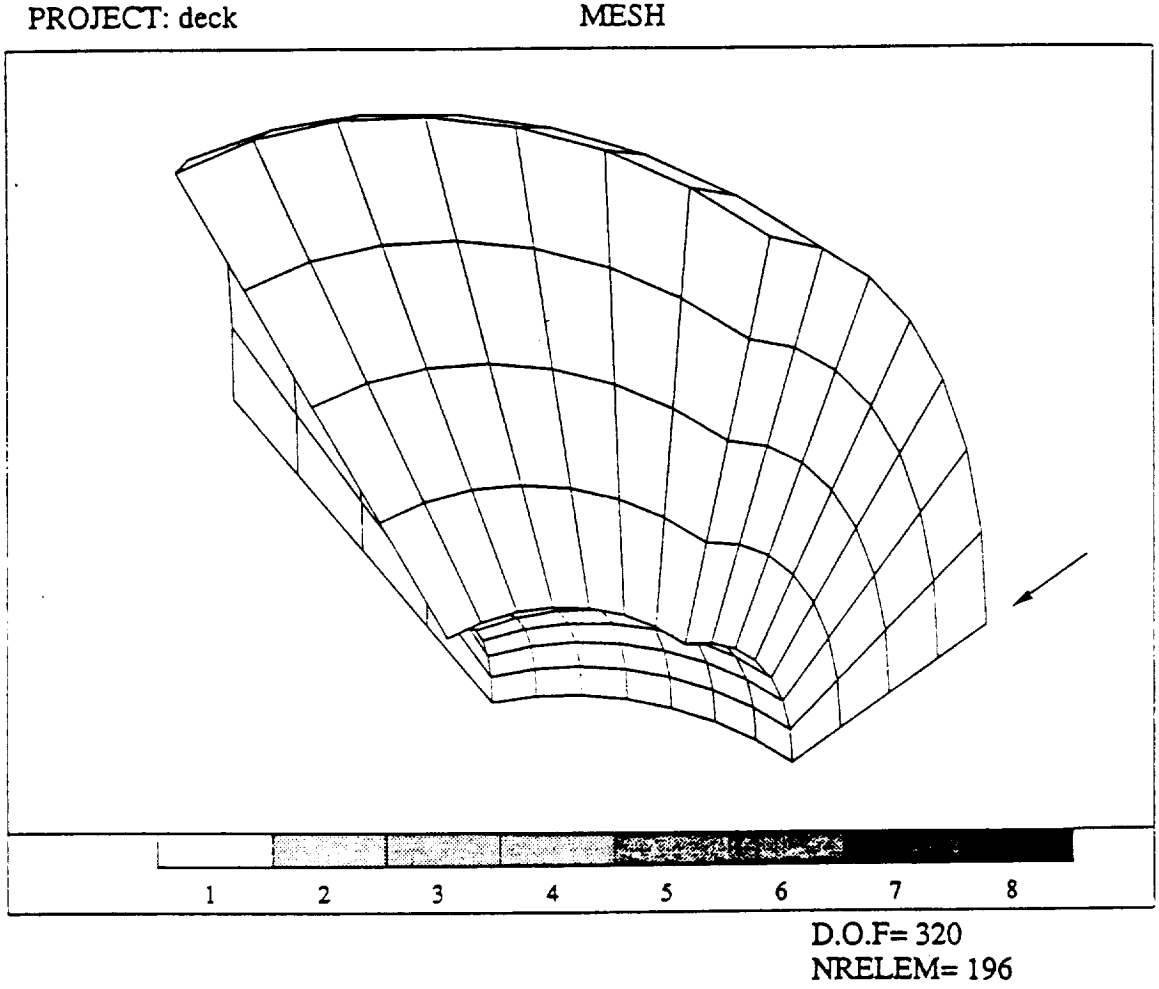


Figure 7.49: Flow around a sphere, $M = 6$. An initial mesh.

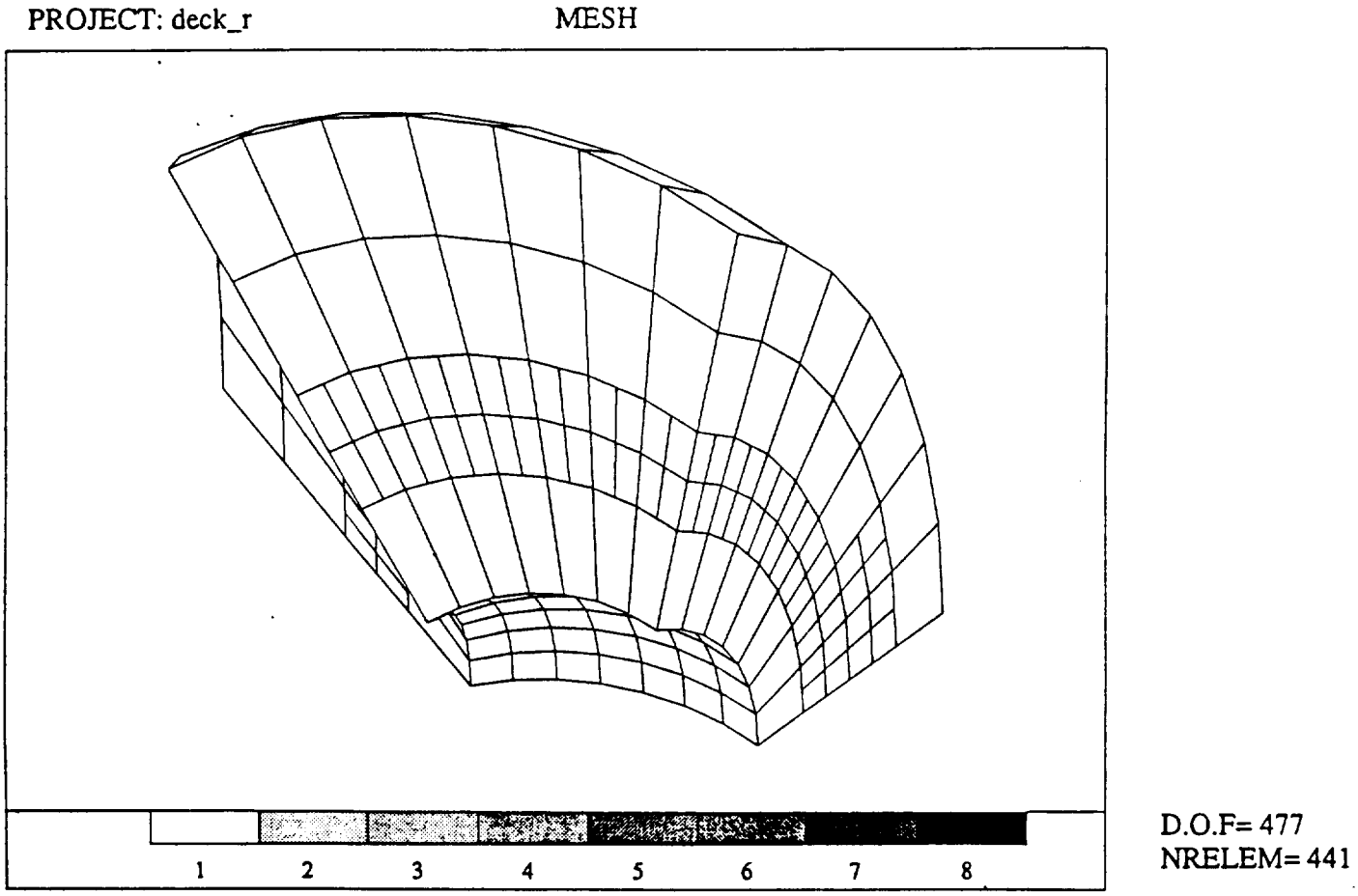


Figure 7.50: Flow around a sphere, $M = 6$. h-adaptive mesh.

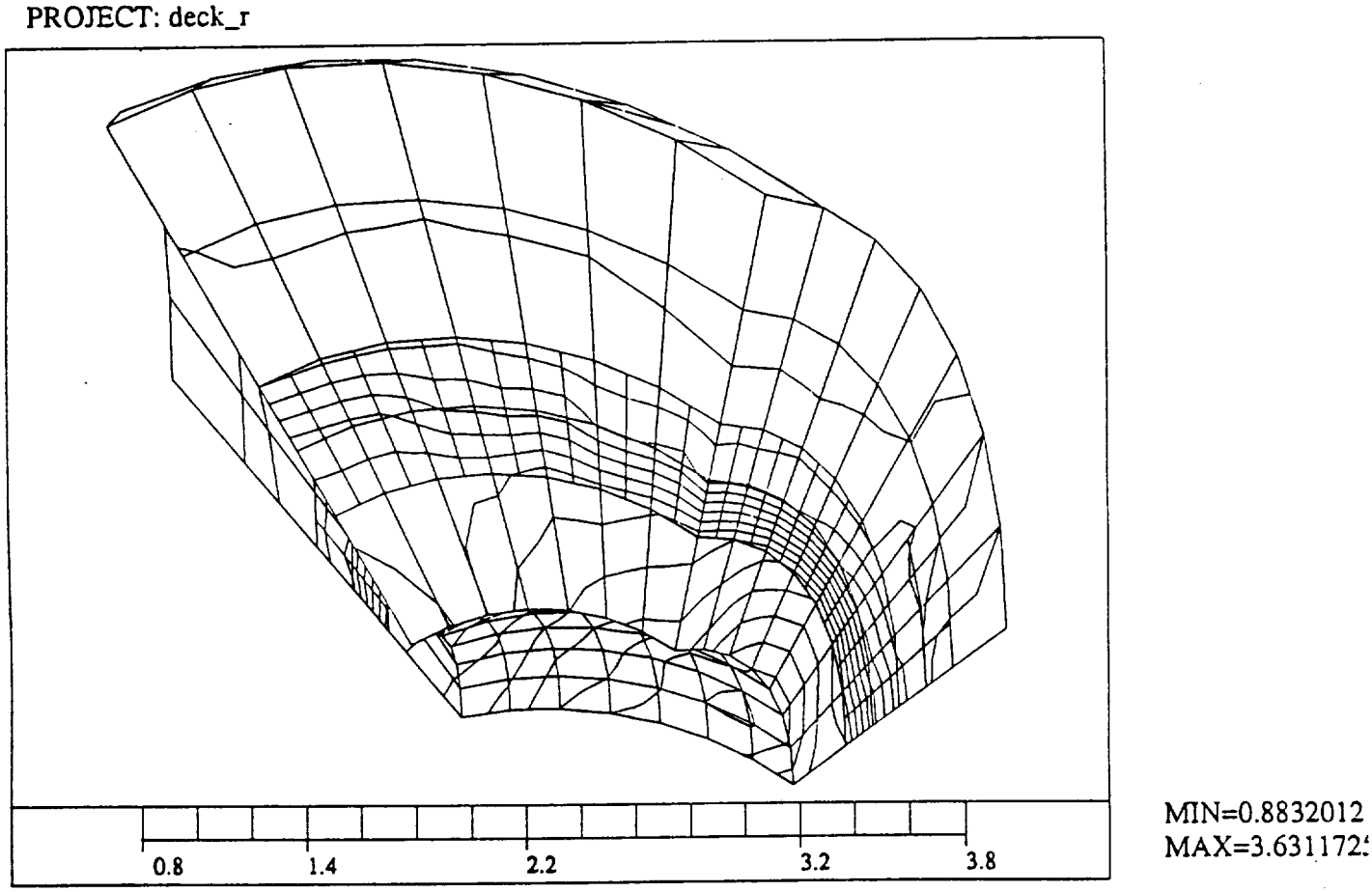


Figure 7.51: Flow around a sphere, $M = 6$. Density contours.

After converging to a steady state solution on this mesh h-adaptations were performed based on interpolation error indicators. The new mesh and density contours are shown in Figs. 7.53 and 7.54. Observe that major features of the flow, the shock and boundary layer zone, are well–resolved. Figs. 7.55 and 7.56 present plots of the skin friction coefficient and the heat flux coefficient.

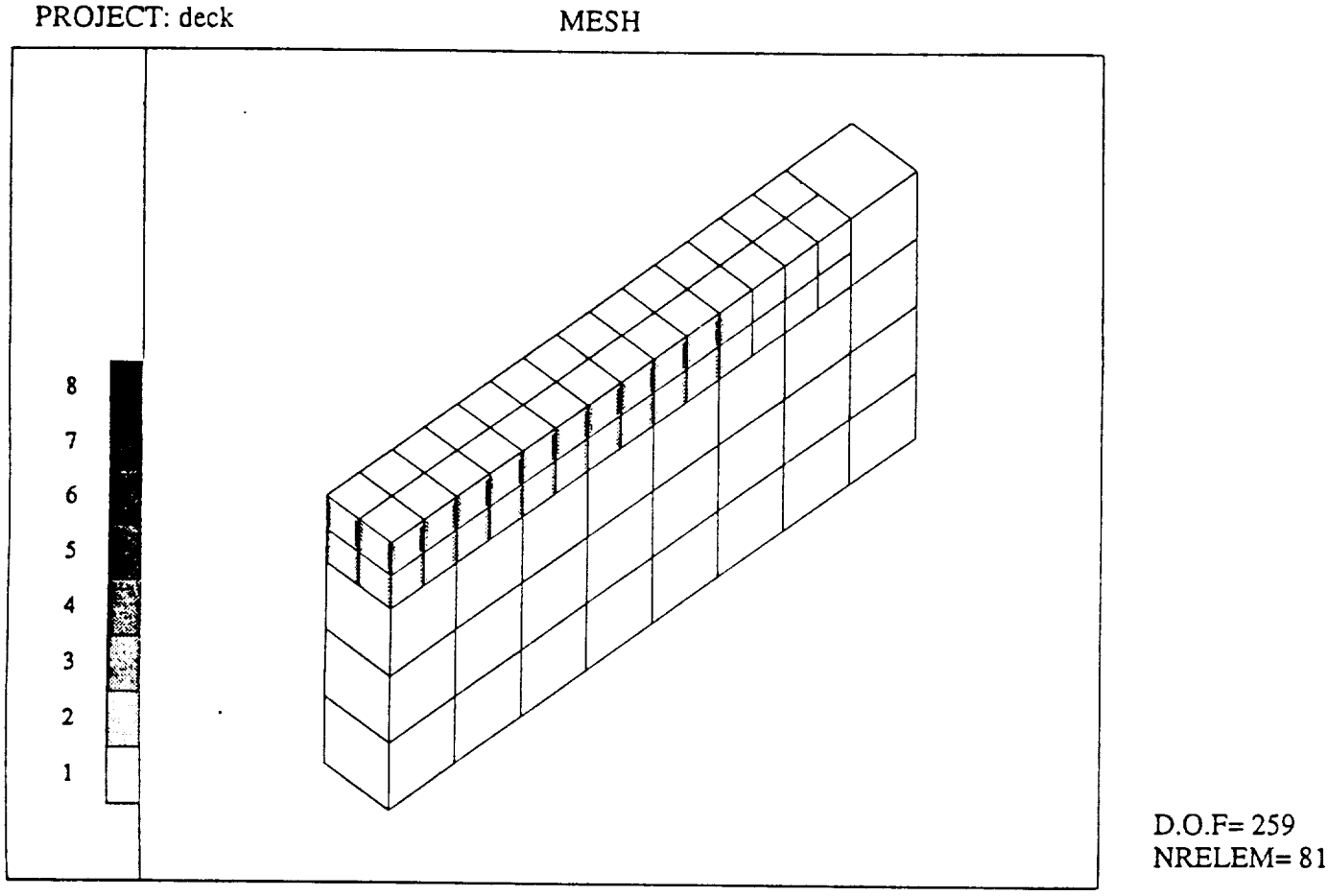


Figure 7.52: Viscous flow over a flat plate, $M = 3$. h-p adapted mesh after one refinement iteration.

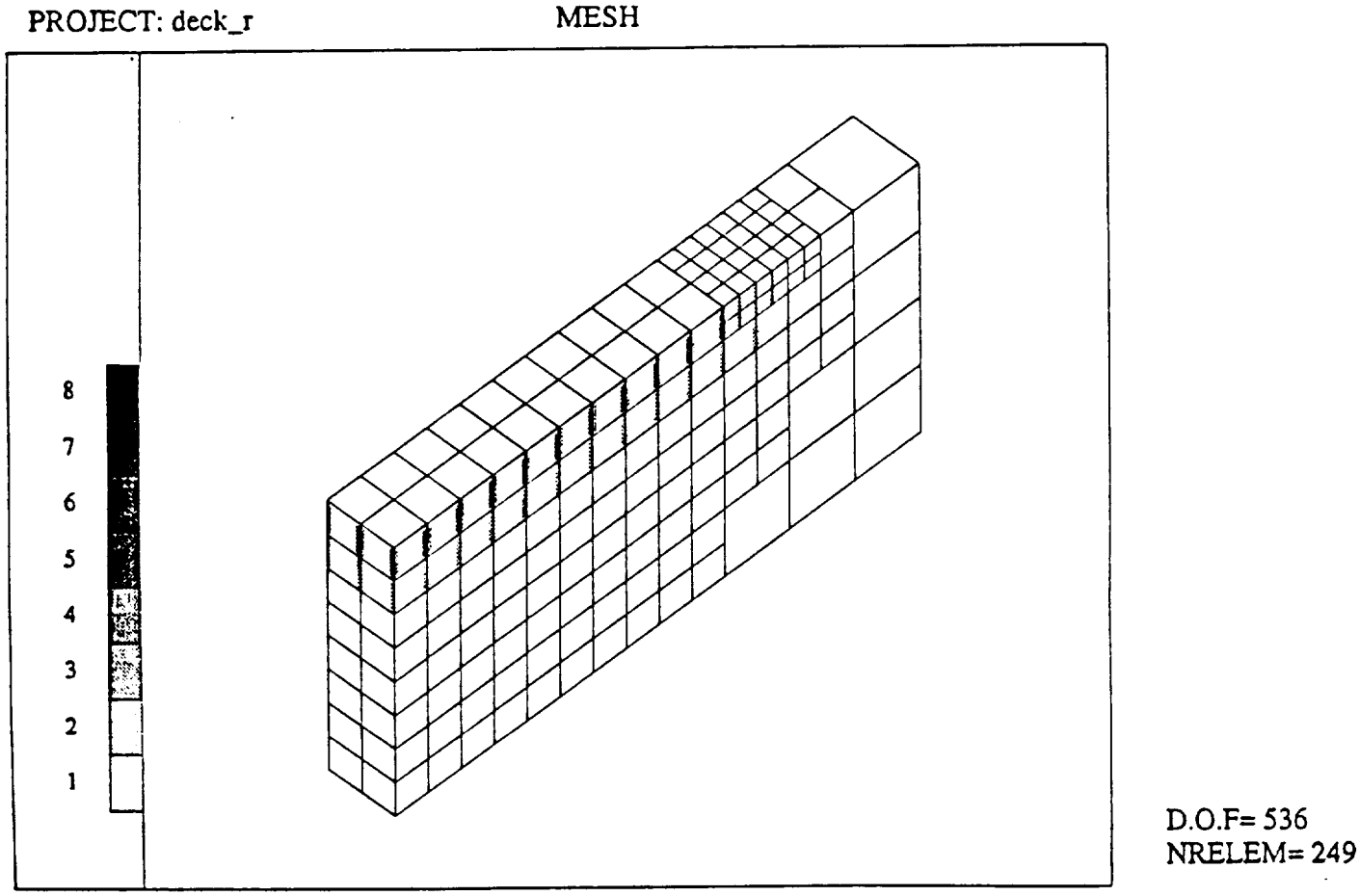


Figure 7.53: Viscous flow over a flat plate, $M = 3$. h-p adapted mesh after two refinement iterations.

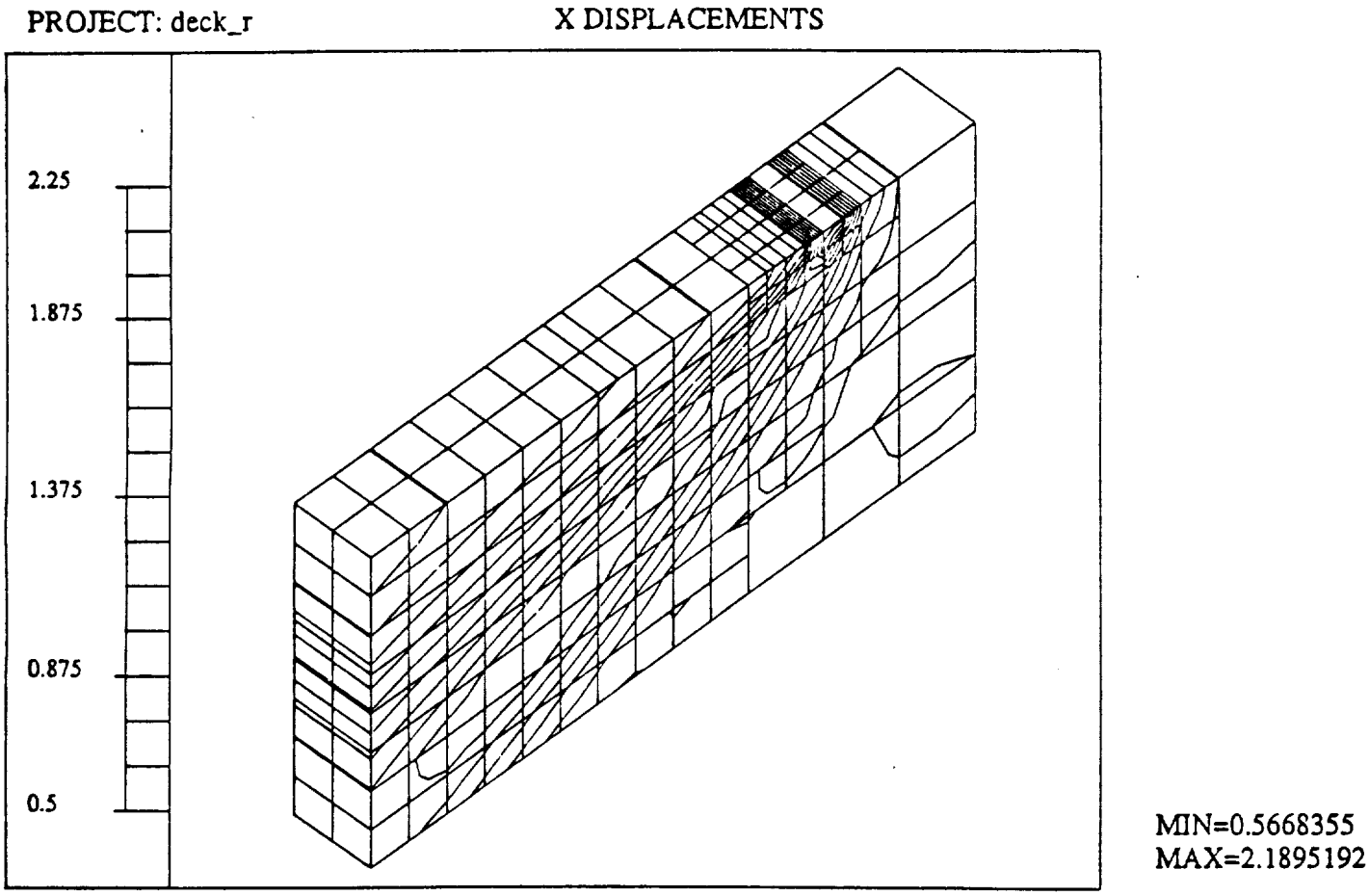


Figure 7.54: Flat plate problem, $M = 3$. $Re = 500$. Density contours.

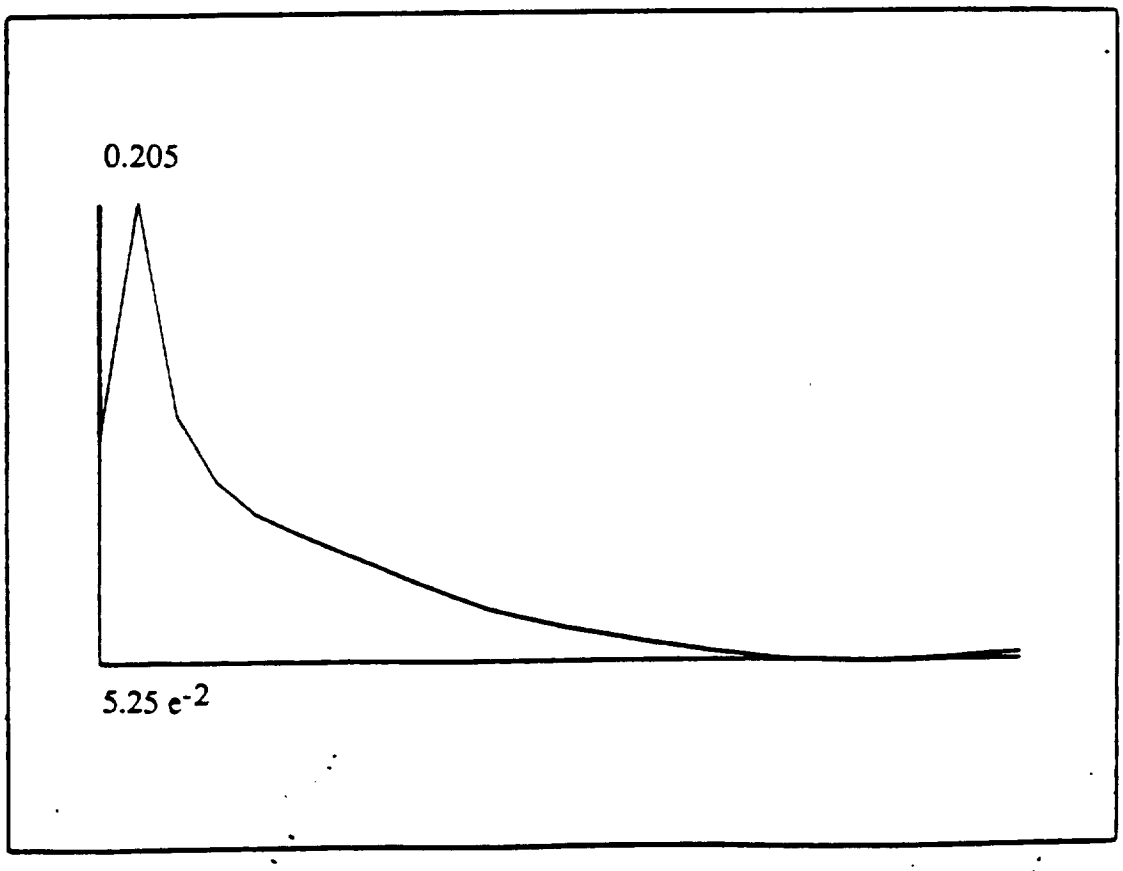


Figure 7.55: Carter flat plate, $M = 3$. Profile of skin friction coefficient along the plate.

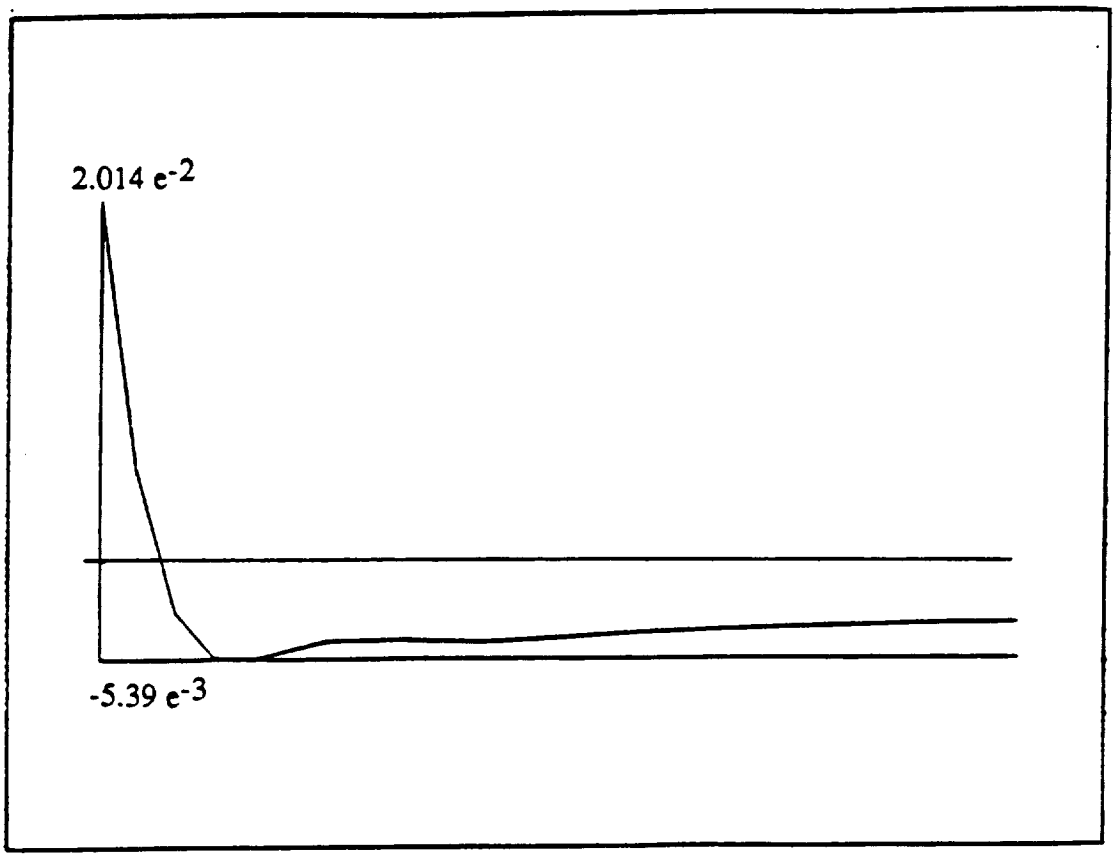


Figure 7.56: Carter flat plate, $M = 3$. Profile of the heat flux coefficient along the plate.

Benchmark Problems

Example 9: Blunt Body Problem With a Type IV (Edney) Interaction

The first benchmark problem is a supersonic viscous flow around a blunt body problem with an incident shock, as defined in Fig. 7.57 and described in [32]. During the first phase of this analysis we applied the adaptive finite element method to the *inviscid* case. A set of preliminary results for this case is presented here in Fig. 7.58 to Fig. 7.62. The initial mesh consists of 32 by 16 linear elements and is shown in Fig. 7.58. The final h-adapted mesh with 3 levels of refinement and the contours of density are shown in Fig. 7.59 and 7.60, respectively. The major deficiency in this solution is a poor resolution of the shock near the cylinder, where the error estimator (the residual technique) was not indicating large errors, at least as compared to the other shocks. To improve the quality of the solution, an additional manual mesh refinement was introduced in this region simultaneously with the automatic mesh adaptation procedure. A solution on this final mesh, Fig. 7.61, is shown in Fig. 7.62. (The above results was obtained by the two-step procedure described in Sec. 2.)

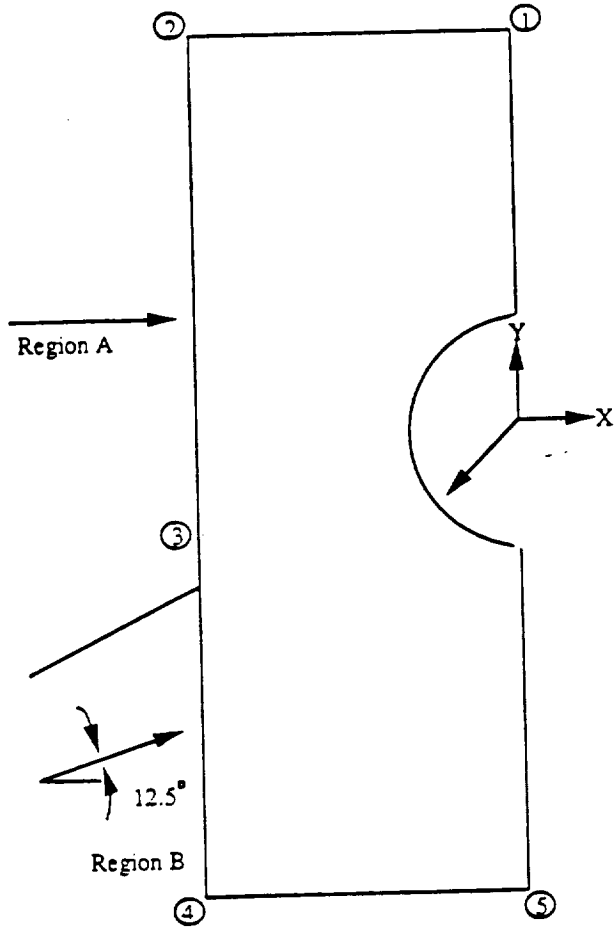
During the second phase of the analysis, the work has focused on the *viscous* solution of the blunt body problem, where the freestream Reynolds number (based on the cylinder radius) was $2.00977*10^5$, and the wall temperature was fixed at $530°R$. Two different meshes have been used to solve this case. The first mesh (referred as mesh A), shown in Fig. 5.7, consists of 40 by 18 linear elements initially, and is graded by a geometric progression of factor 1.01 in the radial direction. To initialize the flow, the inviscid solution was calculated on a one-level uniformly refined mesh. This solution was used as a starting point for a viscous analysis. The mesh with two levels of h-adaptation and the pressure contours of the viscous solution are presented in Figs. 5.8 and 5.9. A 3D perspective view of the pressure is also shown in Fig. 7.63. While the result shows a reasonable shock resolution, the viscous features near the cylinder are still far from fully resolved and do not compare well with experiment data provided by NASA LaRC. This indicates a need to introducing more degrees of freedom along the cylinder to resolve the viscous phenomena. Notice that the smallest thickness of the element along the cylinder after two levels of refinement is still greater than 1% of the radius. Ref [32] also indicates that the dominant viscous region is less than 1% of the radial distance from the cylinder.

In order to resolve viscous phenomena near the wall, a higher order approximation was introduced in the boundary layer. A mesh (referred to as mesh B) with such features is shown in Fig. 7.64. The boundary layer zone of width approximately 0.01 was covered by nine layers of second order elements with sizes geometrically graded toward the wall (with ratio $q = 0.5$). The size of the smallest element is was based on laminar boundary layer theory as well as on results presented in literature [32]. The rest of the computational domain is discretized by

40 by 18 linear elements. A final adapted mesh with two levels of h-adaptation (with 7082 degrees of freedom) is show in Fig. 7.65. The corresponding pressure contour and sonic line are presented in Figs. 7.66 and 7.67, respectively. The 3D perspective views of the density and pressure in the shock–boundary layer interaction area are shown in Figs. 7.68 and 7.69, where the embedded shocks in the jet zone can be clearly identified.

The mesh with quadratic elements in the near-wall region for this case provided a much better resolution of viscous phenomena and favorable comparisons with experimental results. In particular, the comparison of pressure and heat flux along the cylinder with the experimental data are profiled in Figs. 7.70 and 7.71, respectively. The predicted wall pressure distribution is in very good agreement with the experiment results, except the small difference of the shock impingement location. (The same discrepancy was also reported and explained in Ref [18], where it was essentially attributed to fluctuation in the experimental flow pattern.) The plot of the heat flux shows a good correlation with experimental observations—better than many other numerical references with much finer meshes. Note that the oscillatory character of our numerical heat flux is a consequence of a temporary imperfection in our post-processing package—gradients are calculated and plotted at nodes, where their accuracy is the lowest. A better postprocessing algorithm would be to perform the gradient calculations at integration points and then project the solution to the nodes. Also note that a discrepancy in the magnitude of the heat flux distribution between the numerical and experimental data was also reported in Refs. [18,32]. It should be mentioned that our computational grid has a much coarser circumferential resolution than those used in other references. The cylinder was covered by 40 quadratic elements in the circumferential direction (on the adapted mesh). Because of the curvature effects around the cylinder, high accuracy on such a coarse mesh can only be achieved by using high order elements. For example, mesh A, which consists of linear elements only, and with even more than 10,000 degrees of freedom, failed to provide a reasonable resolution of heat flux.

The numerical results on the last adapted mesh B were obtained using the implicit/explicit procedure. Because the stable time step sizes are always restricted by the extremely thin elements in the viscous layer, both fully implicit and fully explicit schemes are more expensive than the mixed implicit/explicit procedure. For this case, we set the maximum CFL number to 50, and were able to gain the cost reduction factor of 0.127 comparing with fully implicit algorithm. Recall that the cost reduction factor was defined in Section 5 as the ratio of the cost marching in time (and converging to steady state) between the implicit/explicit and fully implicit algorithm. Thus, the above factor indicates that the implicit/explicit algorithm was about 8 times faster—a considerable speedup.



Coordinates		
Point	X/R	Y/R
1	0.0	4.5
2	-2.5	4.5
3	-2.5	-0.4
4	-2.5	-3.5
5	0	-3.5

Flow Conditions

	Region A	Region B
Mach Number	8.03	5.25
Density (slugs/ft^3)	4.866×10^{-5}	16.198×10^{-5}
Temperature ($^\circ$R)	220.0	469.43

Figure 7.57: Blunt body problem, initial conditions and geometry.

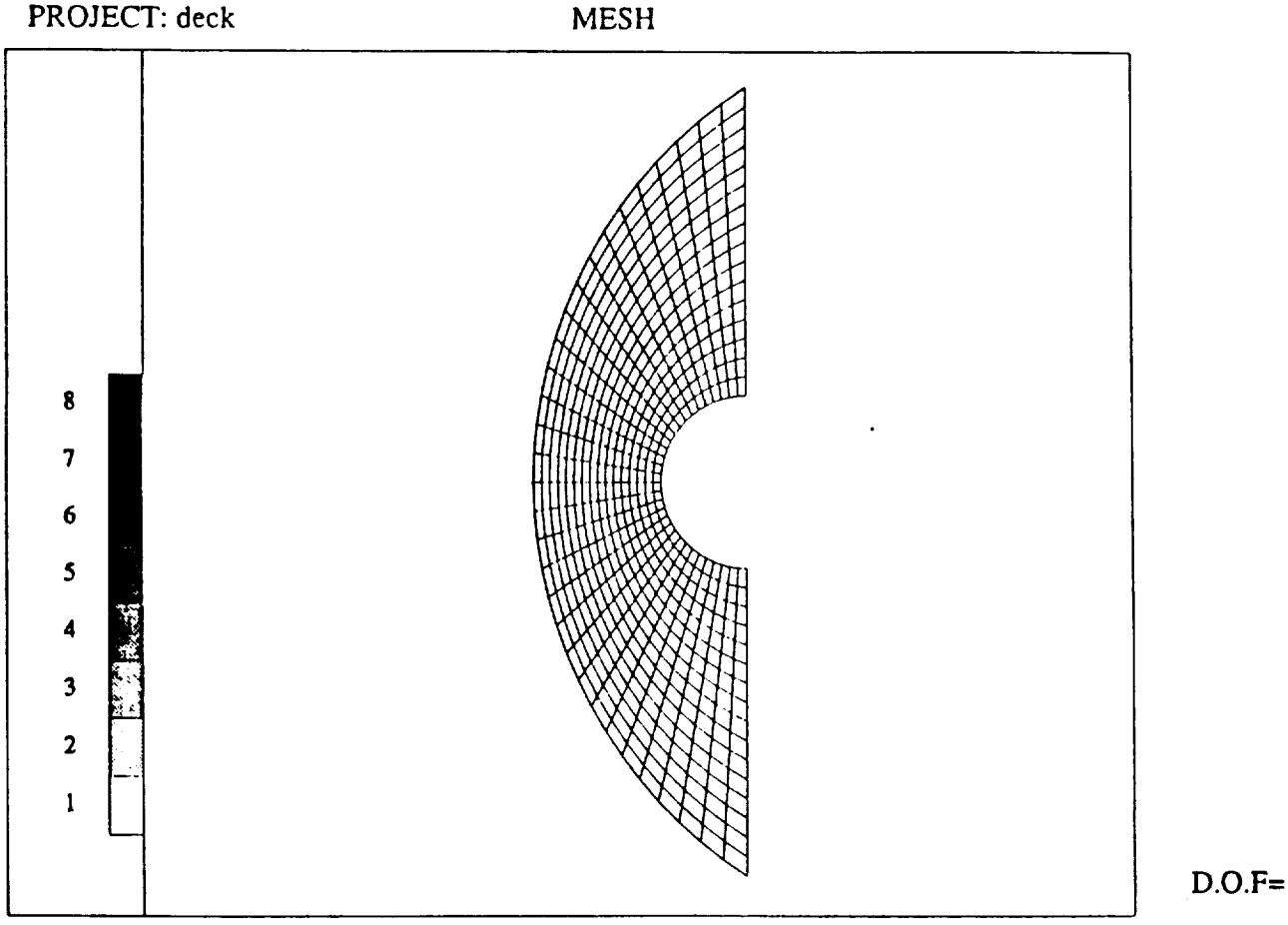


Figure 7.58: Blunt body problem, initial computational mesh.

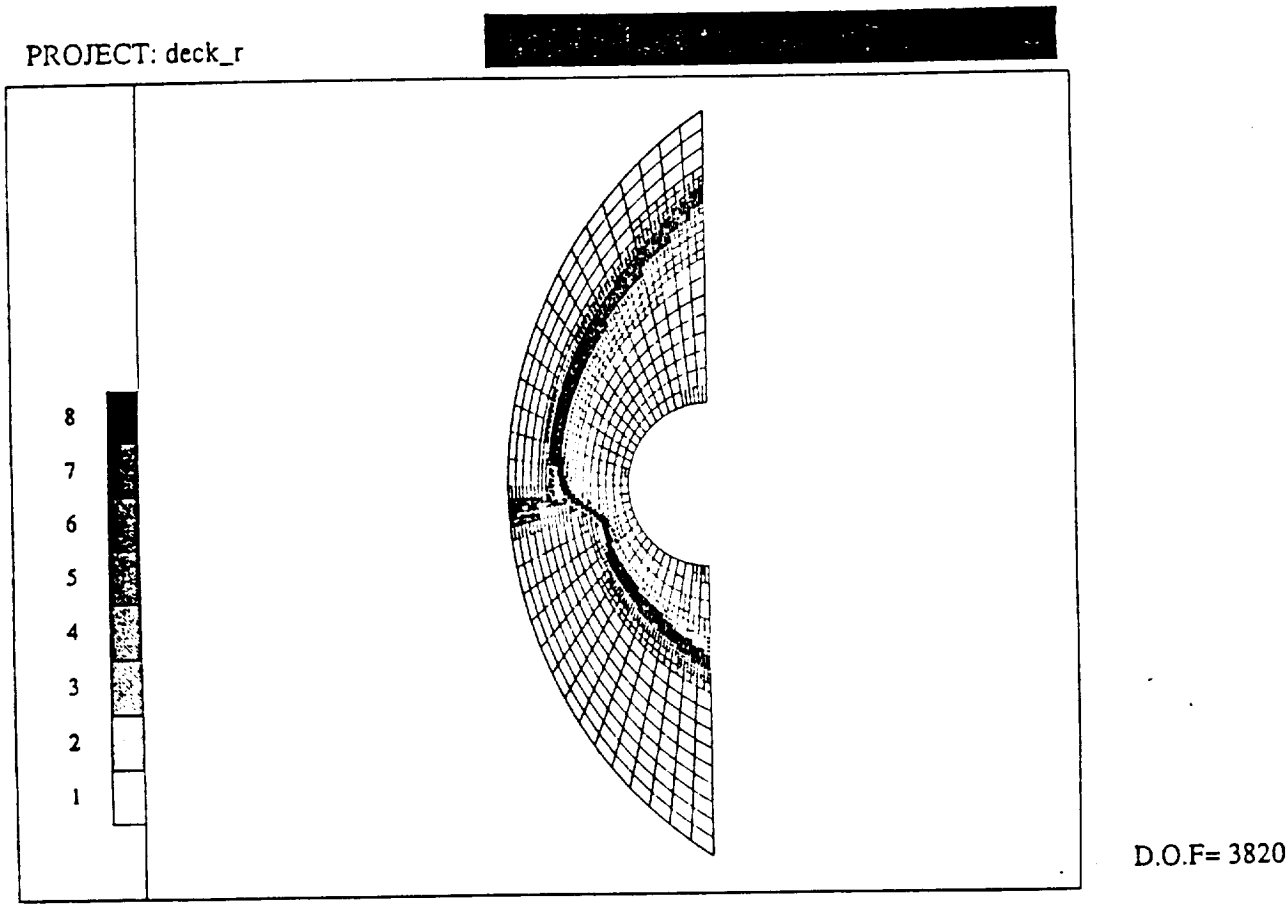


Figure 7.59: Blunt body problem, automatically h-adapted mesh used in the inviscid analysis.

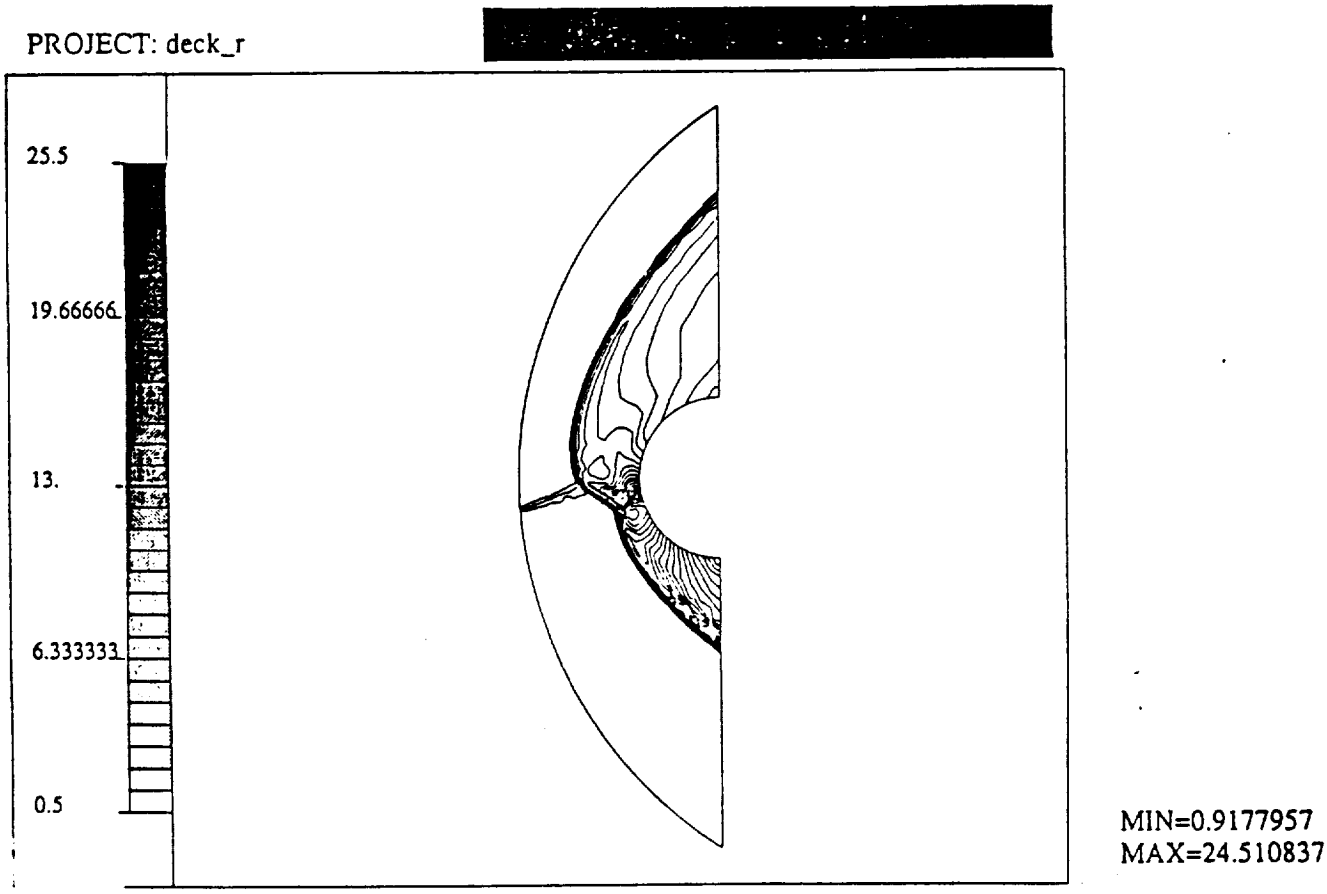


Figure 7.60: Blunt body problem, density contours for the inviscid solution.

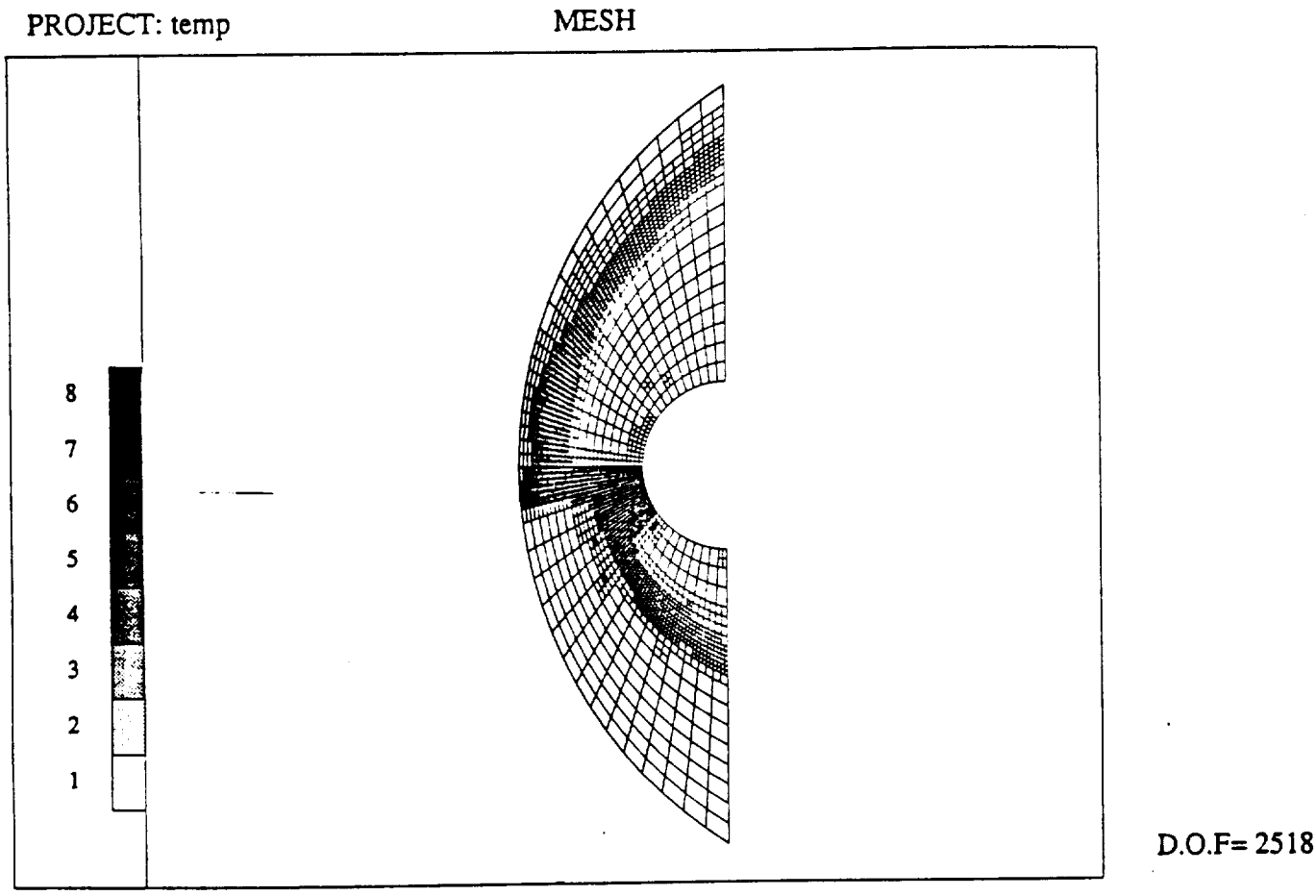


Figure 7.61: Blunt body problem, final mesh with additional manual adaptation in the near wall region.

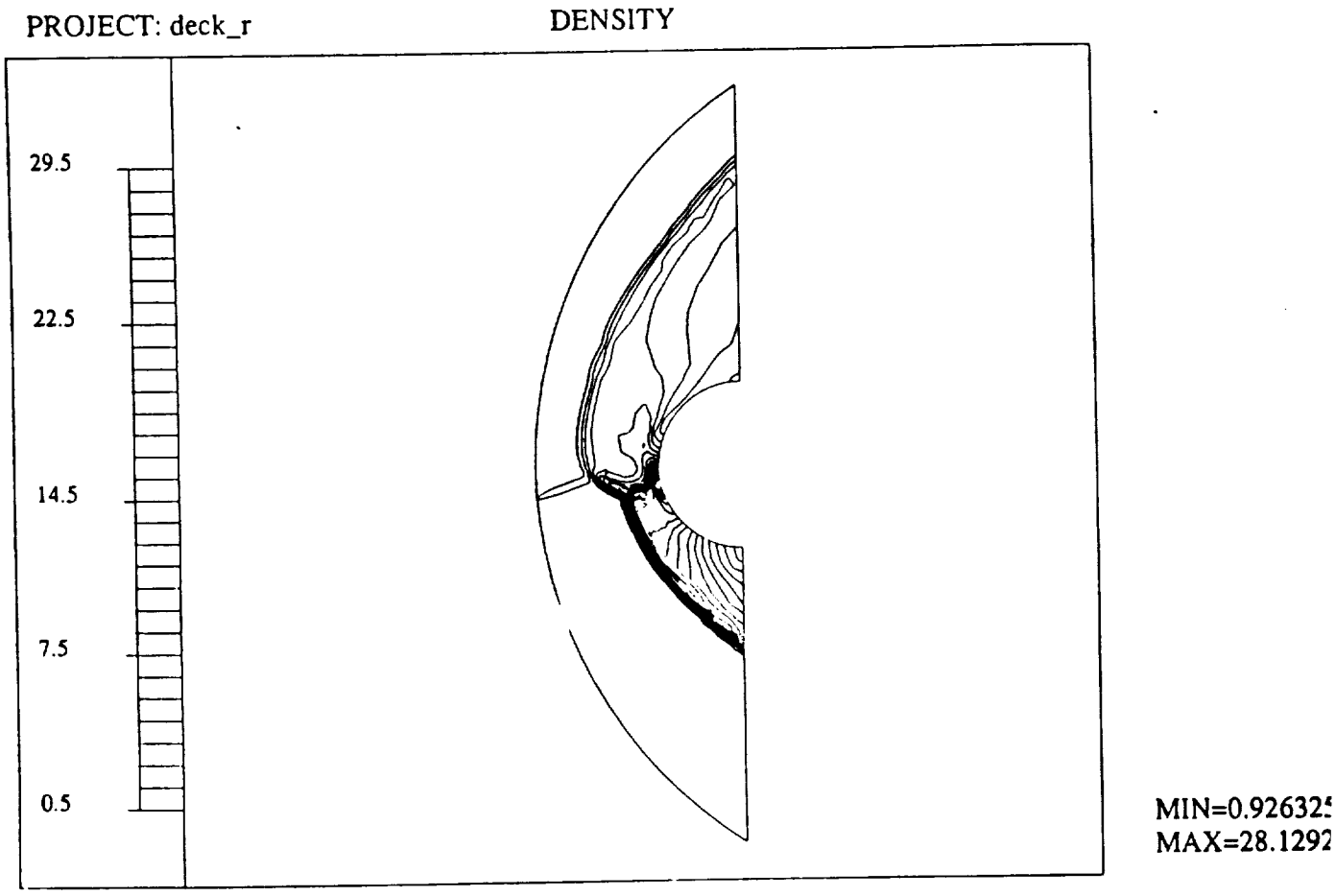


Figure 7.62: Blunt body problem, density contours for the final mesh.

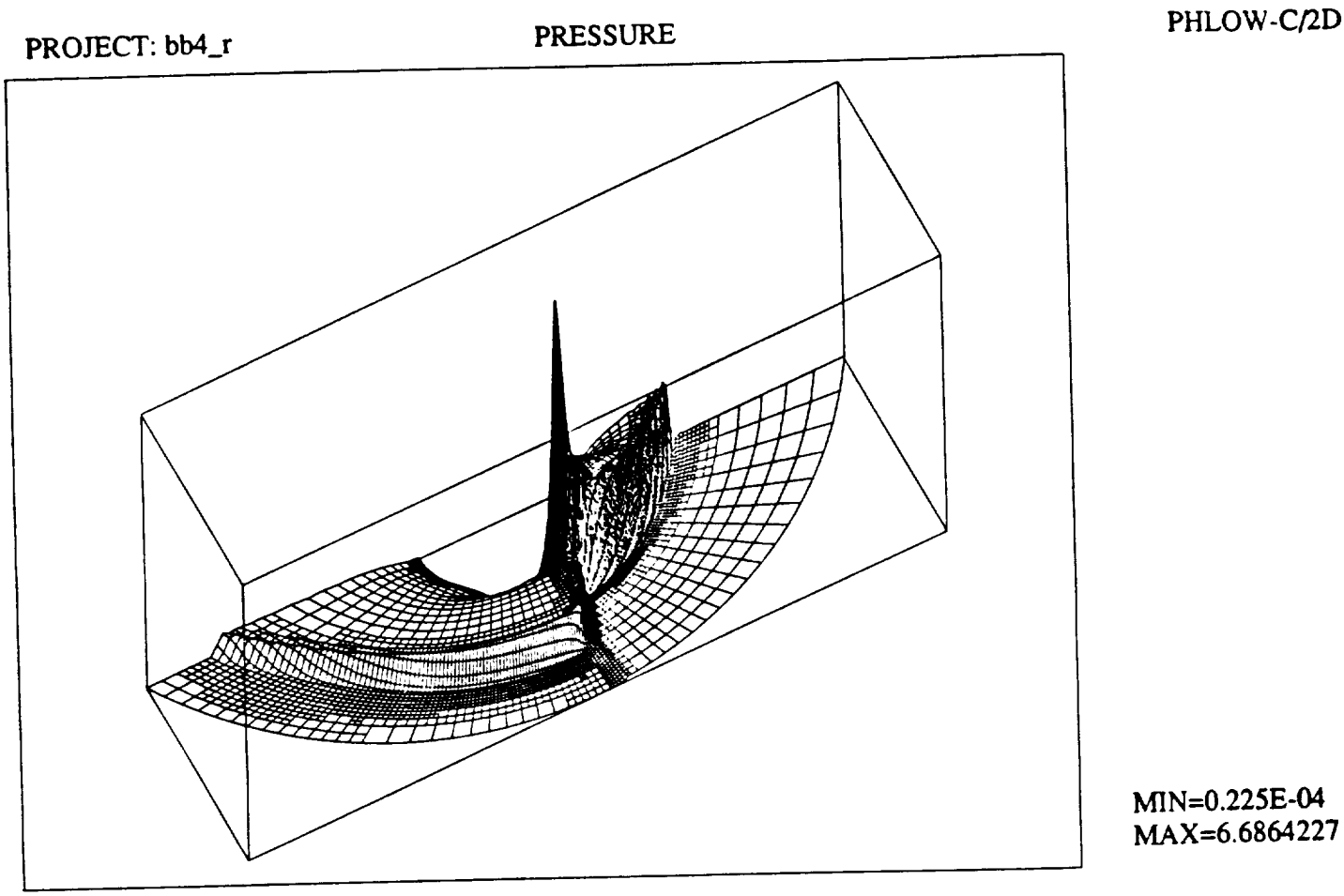


Figure 7.63: Blunt body problem, 3D view of pressure.

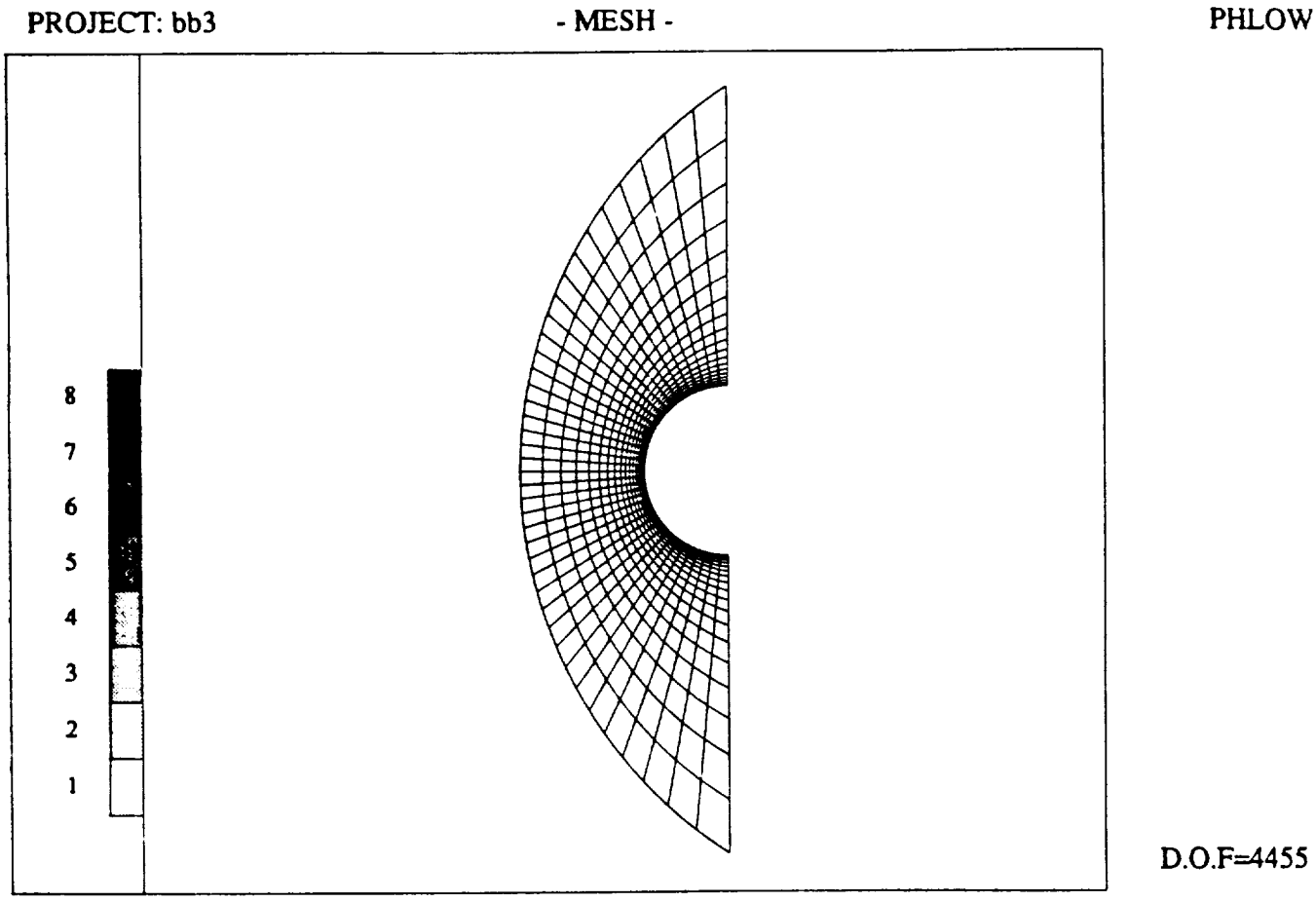


Figure 7.64: Graded Mesh, includes 9 layers of quadratic elements along the cylinder (Referred as Mesh B.)

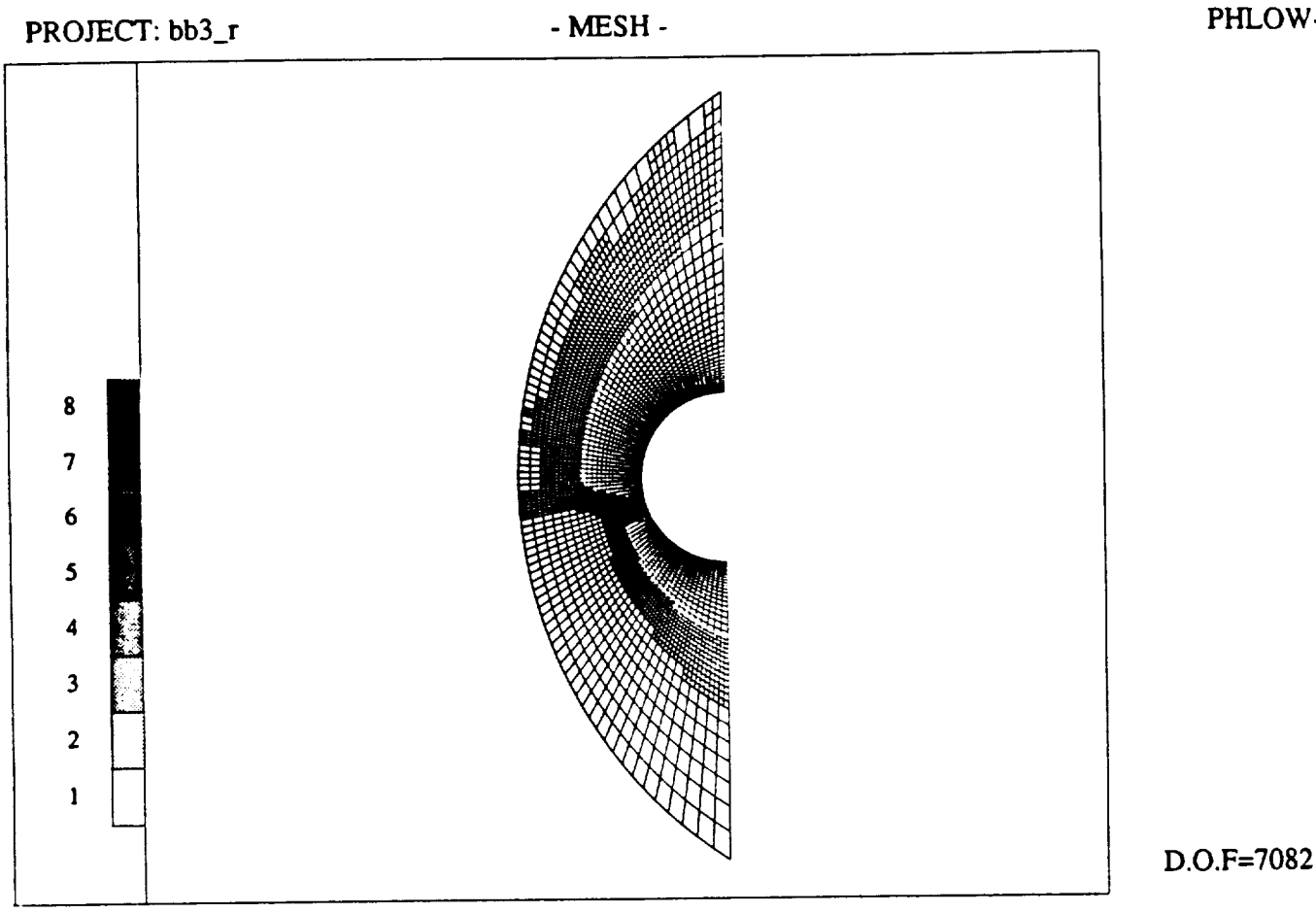


Figure 7.65: Mesh B after 2 levels of adaptation. Elements on the wall are of the second order.

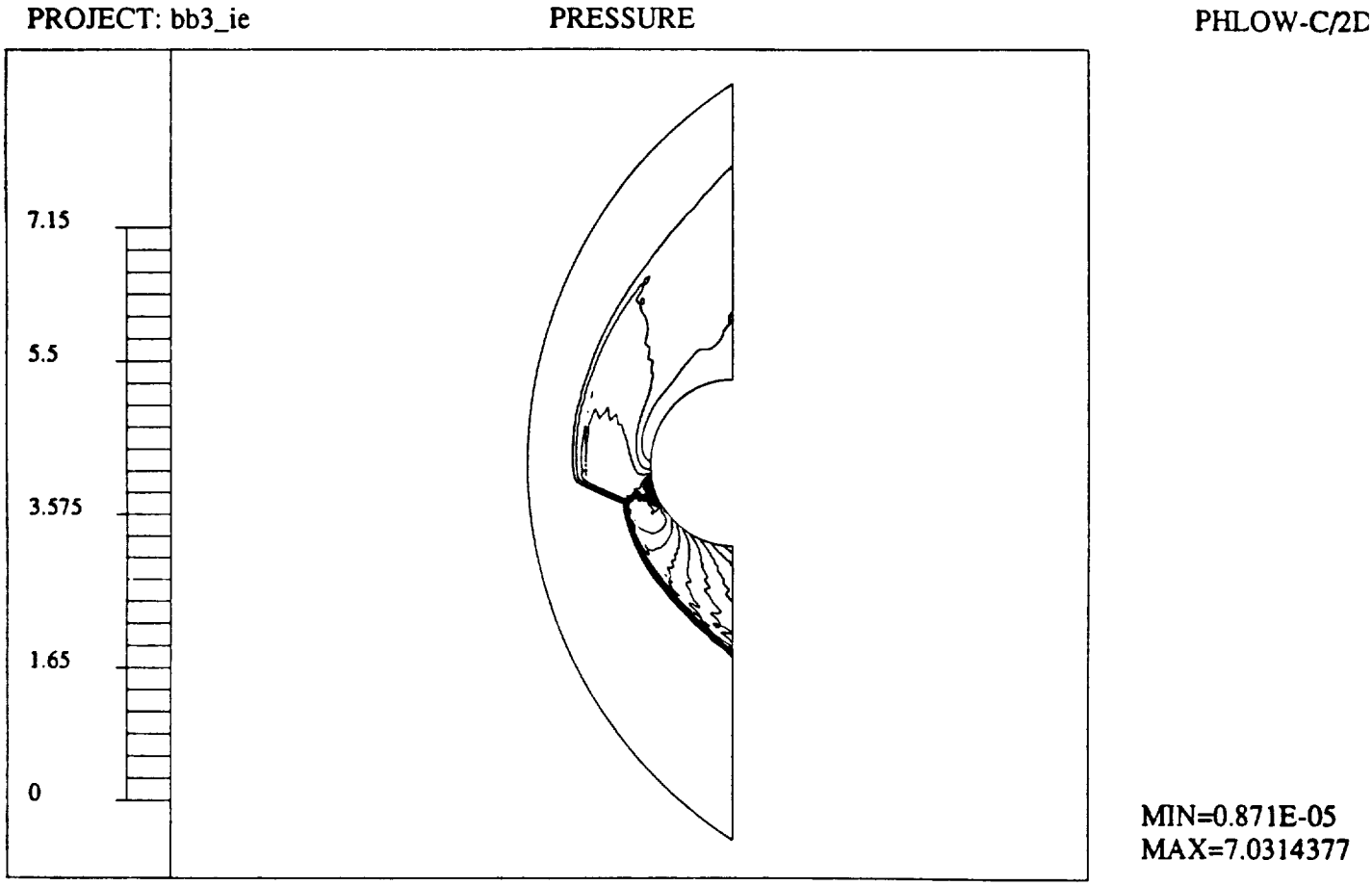


Figure 7.66: Pressure contours obtained on the adapted Mesh B.

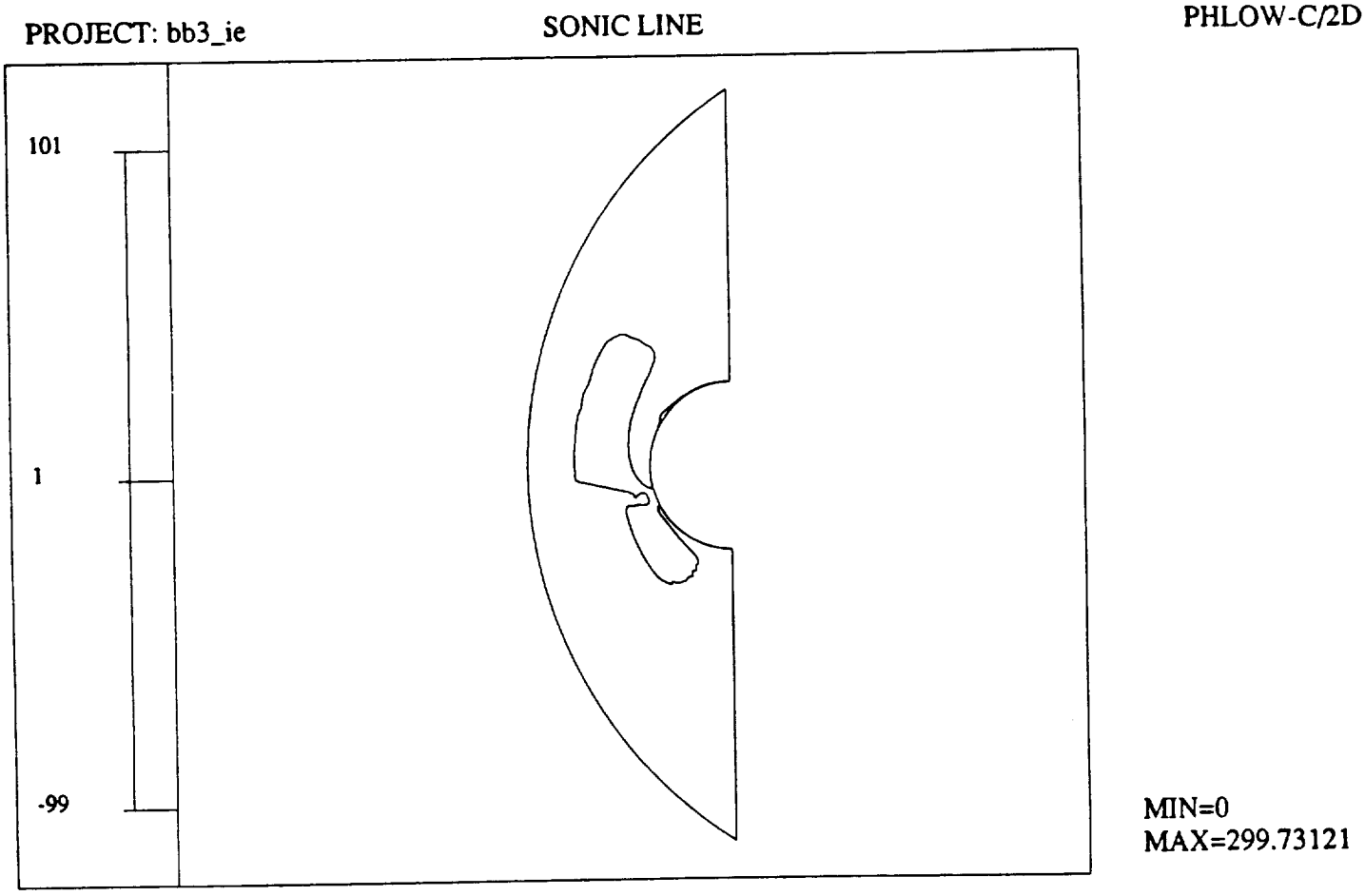


Figure 7.67: Sonic line obtained on adapted Mesh B.

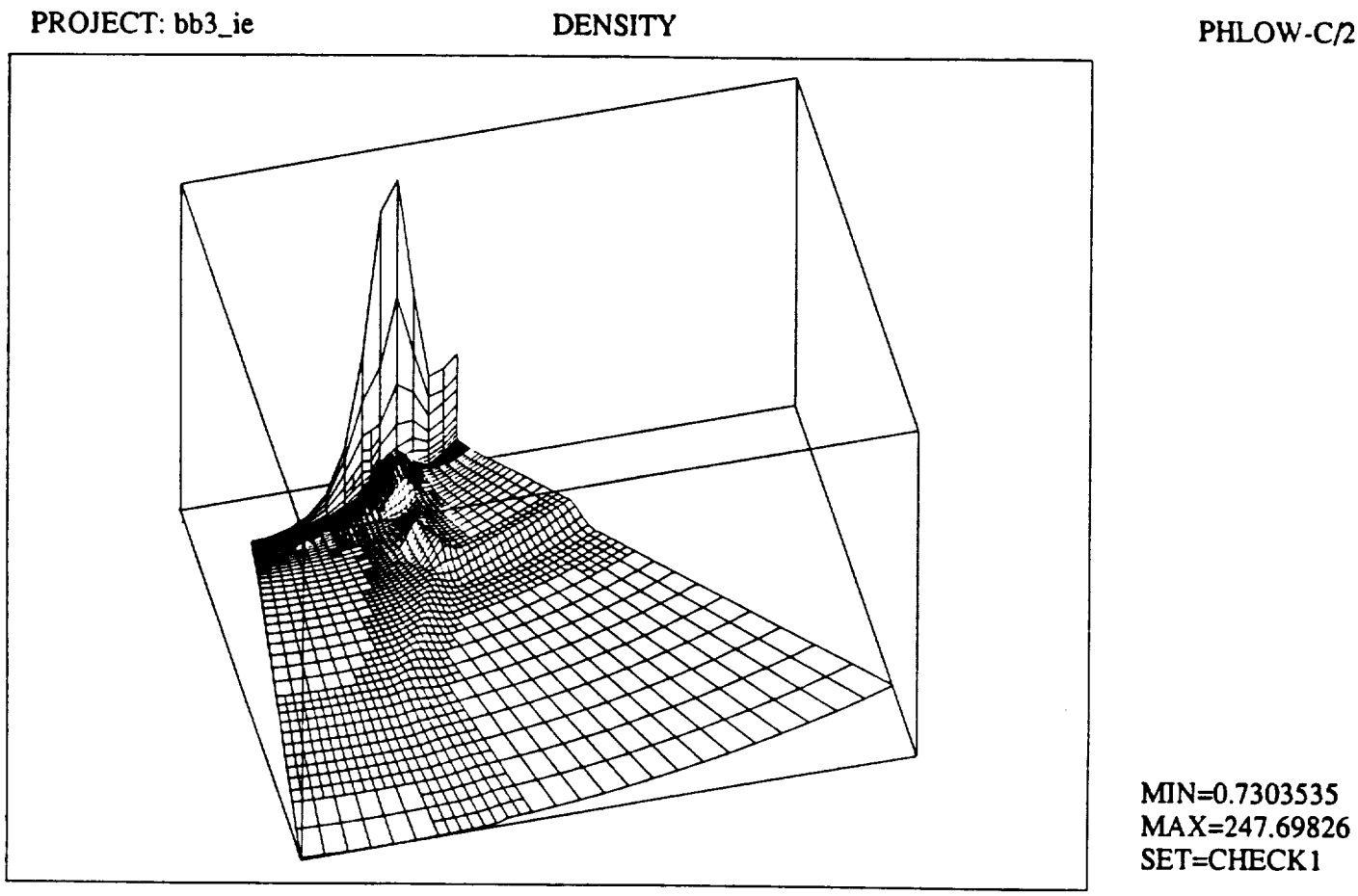


Figure 7.68: 3D view of density in the shock impingement region.

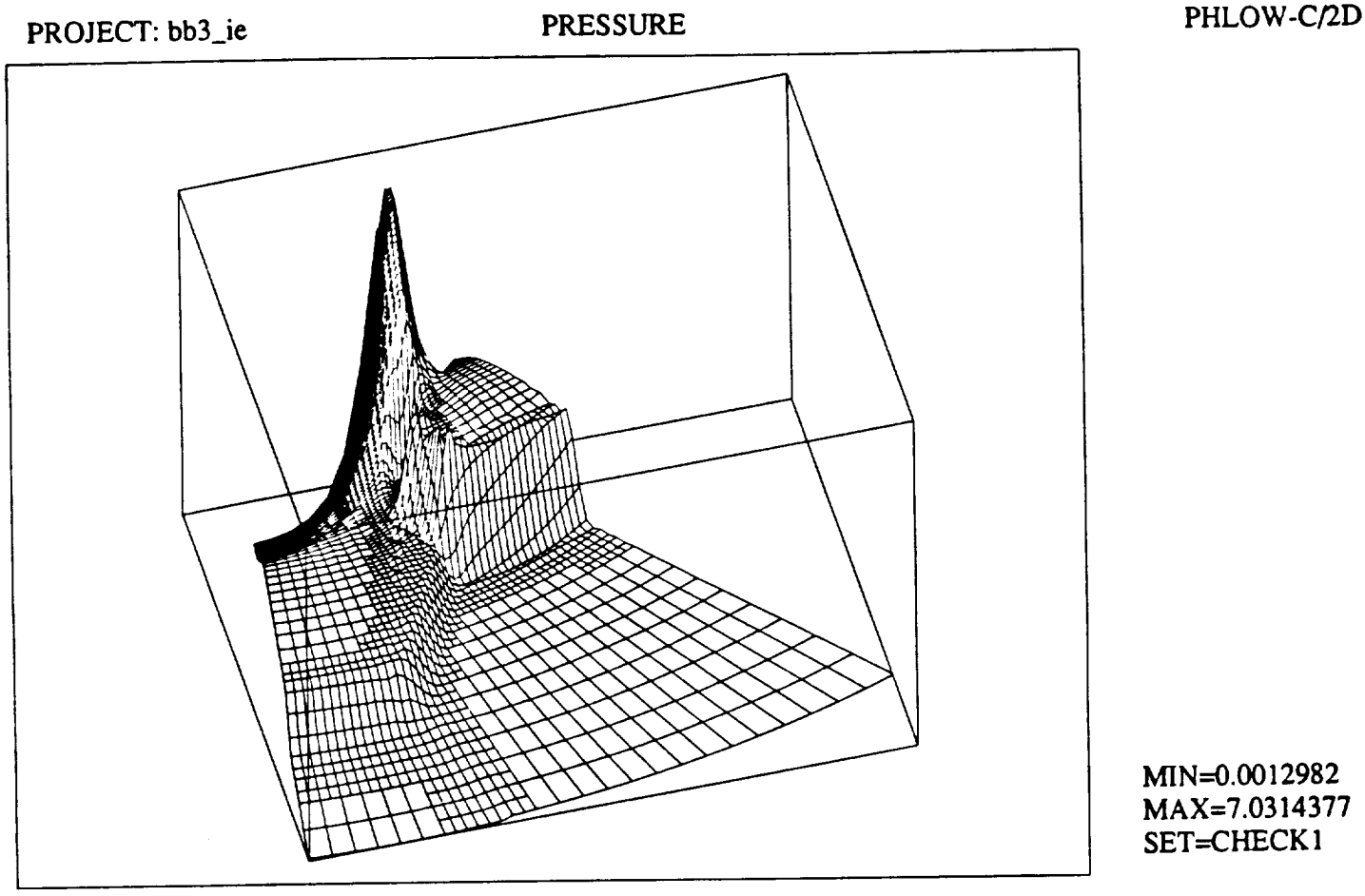


Figure 7.69: 3D view of pressure in the shock impingement region.

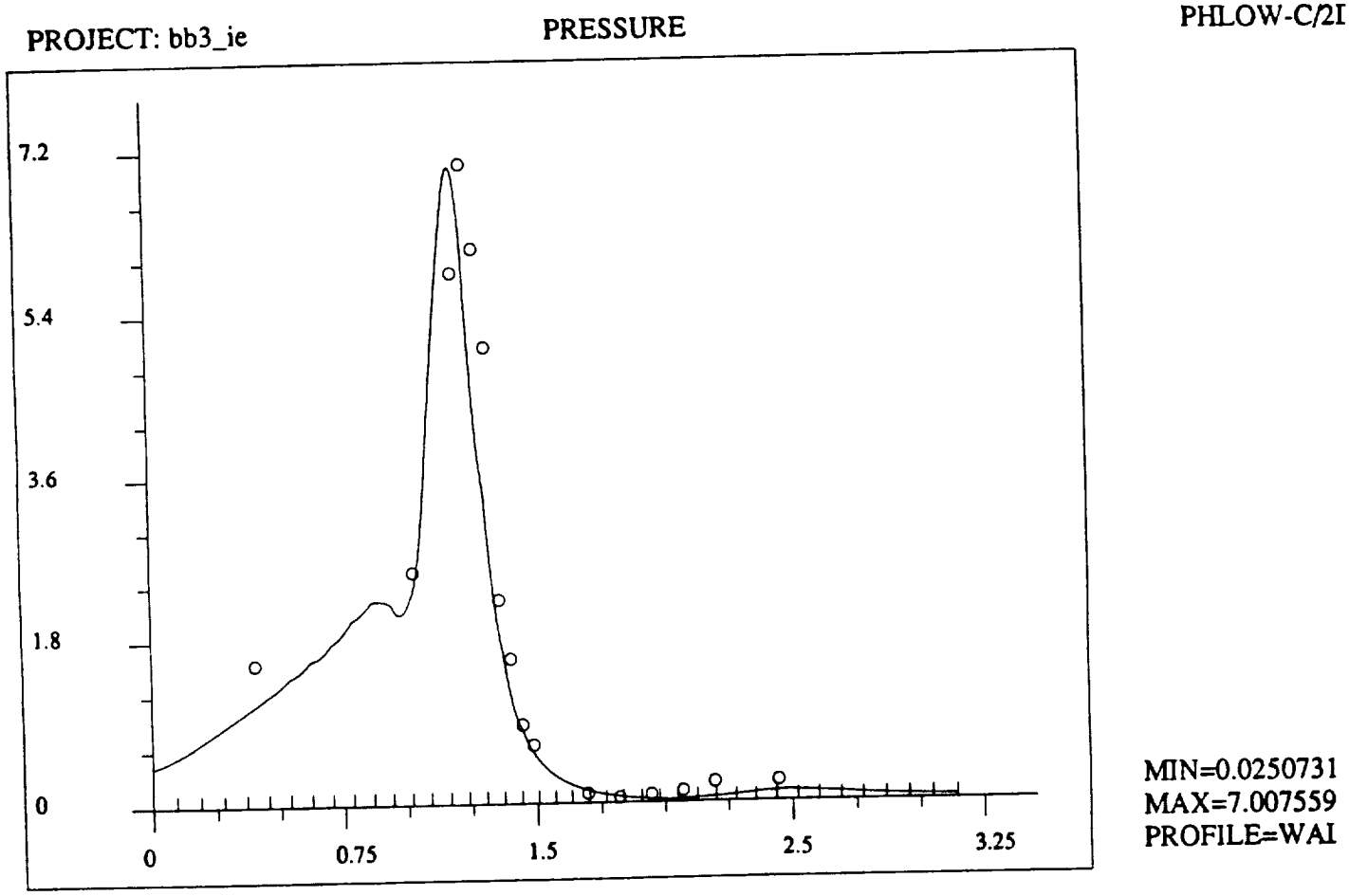


Figure 7.70: Wall pressure distribution.

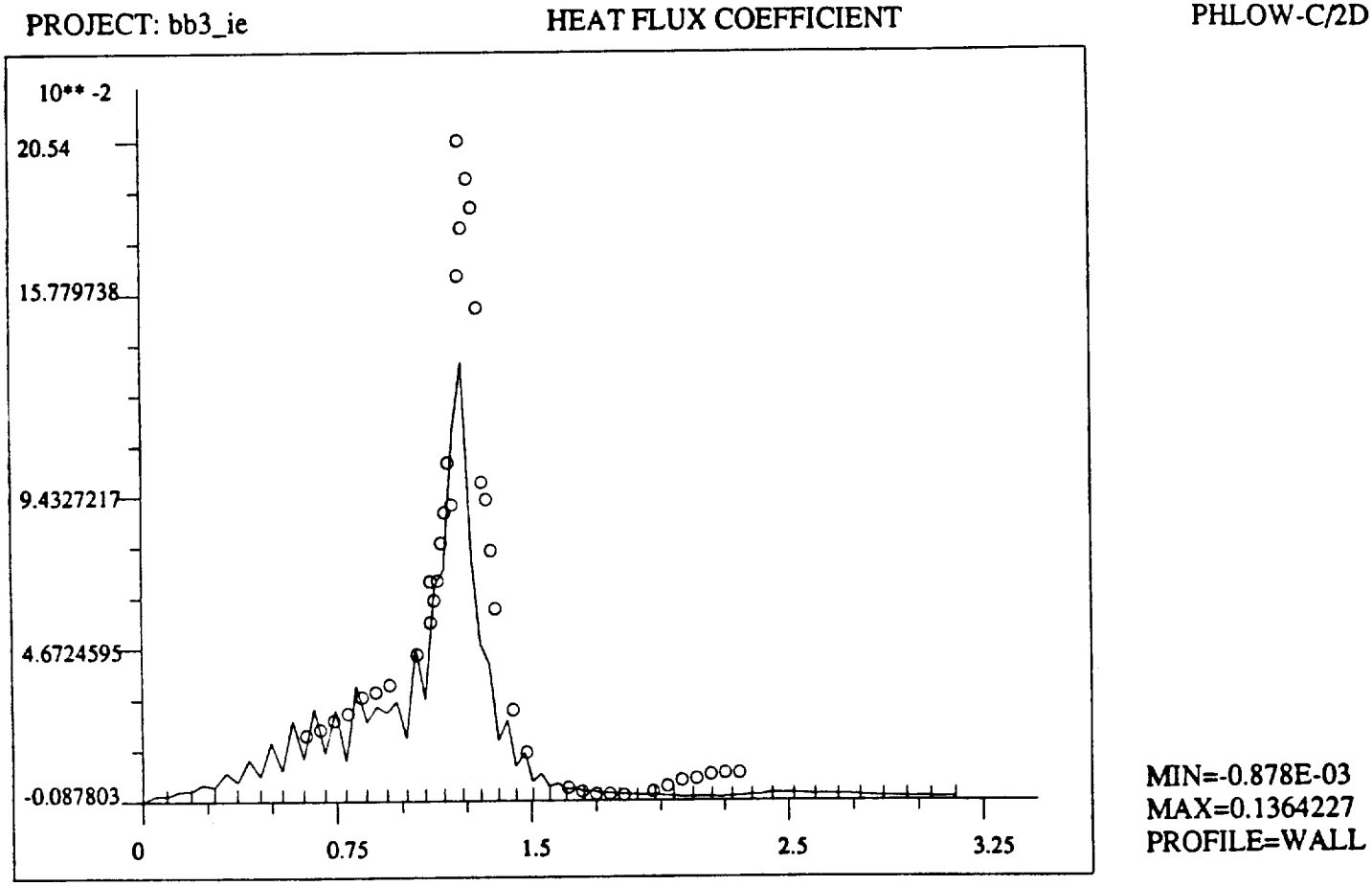


Figure 7.71: Wall heat flux distribution.

Example 10: Shock/Boundary Layer Interaction With Separated Flow

The second benchmark problem involves a shock/boundary layer interaction with separated flow (Holden problem [30,32]). The particular flow conditions used in the analysis of the Holden problem are as follows:

$$\begin{aligned} M &= 14.1 \\ Re &= 72,000 \\ T_\infty &= 80\,°K \\ T_{\text{wall}} &= 297.2\,°K \end{aligned}$$

inclination of the wedge = 24°

We initiated work on this problem in the previous phase of the project, but encountered considerable difficulties in resolving the flow recirculation region. These difficulties were resolved in the Phase II effort and, moreover, the source of our previous difficulties was identified as the artificial dissipation model. The effort spent on solving this problem during the previous phase is summarized in the next two paragraphs, and the related numerical results are included in Figs. 7.72 to 7.77.

The first approach for solving this problem was with the two-step algorithm using an initial mesh of 11 × 25 linear elements. The elements were stretched in the horizontal direction with the aspect ratio along the solid wall boundary of only about 1:5 and the size of the smallest elements is about 0.03. The problem turned out to be practically unsolvable without the artificial viscosity especially designed for highly stretched or distorted elements. The standard artificial viscosities (Lapidus' and Morgan's models described in Section 2.4) failed to stabilize the solution around the stagnation point. On the other hand, introducing elements with an aspect ratio of 1 in the wall area results in a prohibitively large number of elements and small time step, slowing down the integration process. However with our modified artificial dissipation model the solution process was able to proceed on the original mesh. An *h*-adaptive mesh with three levels of refinement (based on residual error indicators) is shown in Fig. 7.72, the corresponding density contours are shown in Fig. 7.73. Salient points about the solution to note include: sharp shock resolution, minimal number of degrees of freedom to capture the shocks, and reflected shock/leading edge–shock/boundary layer interaction. However, the key viscous feature along the wall, the recirculation bubble, has not been resolved.

In a parallel modeling effort, we also used the one-step Taylor-Galerkin algorithm which allows one to use much larger time steps and therefore leads to faster convergence of the steady state solution. The initial linear mesh used in this solution procedure is shown in Fig. 7.74. Compared to the previous mesh we introduced a considerably finer discretization along the solid wall boundary, the size of the smallest elements is now 0.005 and the maximum aspect ratio is 25. The problem was run with the CFL constant set to 2.5. From the solution

obtained after 100 and 200 time steps, the mesh was h-adapted up to the second level, based on the residual error indicators, see Fig. 7.75. The contours of the density for this mesh and v-component of the velocity are shown in Figs. 7.76 and 7.77. While the mesh also provides a reasonable resolution of the shocks, the recirculation region was still not well developed. For this mesh we also introduced a p-enriched mesh along the solid wall, with the hope for better resolution of recirculation of the flow, but this effort was also unsuccessful.

During phase II of this project, we have experimented with several different artificial dissipation (AD) models. We found that this is the crucial factor in resolving the flow separation phenomenon for this benchmark problem. Recall that in Section 2.4 we have implemented three different AD models: 1) the classical Lapidus, 2) the modified Lapidus by Löhner at el., and 3) our version which is a modification of the second model in order to handle elements of high aspect ratios. As expected, out modified model performs better than the other two models on elements with irregular shapes (high aspect ratios or high distortions), and we have used this model successfully to solve all the test problems, including a compression corner problem with a lower Mach number as described previously. For Holden's problem at Mach 14, however, all three models have difficulty resolving the separation region. Usually, in the solution of problems with strong shock-boundary layer interactions, not only does the mesh have to be well adapted according to the flow features, but the AD is also a key factor. From numerous test runs, we have found that for this problem, resolution of the recirculation bubble is extremely sensitive to the application of AD because of the flat shock angle and sharp boundary layer pattern involved. It is well known that, for viscous compressible hypersonic flows, the ideal AD model should provide just enough dissipation to smooth shock discontinuities without oversmearing, and should be formulated to avoid introducing too much dissipation (relative to the natural dissipation) in the boundary layer region. This means that the AD should have an accurate built-in sensor to control the amount of dissipation and detect the direction in which the AD is to be added. All three models (belonging to the Lapidus family) are using the velocity gradient as sensors, and their success are based on the argument that the maximum change of velocity gradient is perpendicular to shocks so the AD is also added in that direction. In the boundary layer region, since the change of velocity gradient is perpendicular to the solid wall, there will be no AD introduced in the tangential direction which may otherwise put too much "artificial" dissipation to accurately resolve the "real" flow features.

In the formulation of our AD model described in Section 2.4, the unit vector $\boldsymbol{\ell}$ was based on the gradient calculated on the master element as in Eq. (2.95). Noticing that the orthogonality is not preserved under the transformation, we used the $\boldsymbol{\ell}$ vector computed on the original element. With this simple remedy, the recirculation bubble was successfully resolved. However, because it is a less dissipative mechanism than other models, the solution is more oscillatory near the shocks and the leading edge, and smaller time steps were required

to successfully converge. The analysis effort based on these various AD models is presented below.

In order to compare the numerical results with those presented in ref [30], we used an initial mesh consists of 27 by 25 linear elements, see Fig. 7.78, which was clustered almost identically to the SM mesh used therein. After two levels of uniform refinement, the thickness of the smallest elements would be $2.42 * 10^{-5} ft$, which also matches the SM mesh. On the mesh with one level of uniform refinement, we have experimented with three different AD models: (a) the Lapidus, (b) the modified version of Morgan's presented in Sec. 2.4, and (c) our modified model with the fix as described in the previous paragraph. The results are shown in Figs. 7.79, 7.80, and 7.81, respectively, and are displayed by the contours of the v-component of the velocity. While the first two cases failed to resolve the recirculation bubble, the last one clearly indicates the flow separation region. At this point it is important to note, that we do not claim here that this variant of artificial dissipation is ultimately better than others. It rather seems that the other models were too dissipative for this problem and were "wiping out" the fragile separation point. The design of an ultimate artificial dissipation model, which would resolve the shocks without smearing other features of the solution, is still an open challenge in computational fluid dynamics.

The same mesh with two levels of h-adaptation is shown in Fig. 7.82. The numerical results presented in Figs. 7.83 to 7.85 show contour plots of the density, v-component of the velocity, and a 3D view of pressure in the recirculation region. A comparison of the pressure, skin friction, and heat transfer coefficients along the solid wall with the experiment data are profiled in Figs. 7.86 to 7.88, respectively. In general, they show a similar overall agreement with experiment data as those numerical results for mesh SM presented in ref [30], except that our heat flux is underpredicted in the shock reattachment region. We believe that this discrepancy is caused by the AD introduced in this region. Intuitively speaking, a certain fraction of the total dissipation on the wall is "taken over" by the AD model, which tends to reduce the flux contributions from the natural dissipation. Note that, since the amount of dissipation due to the AD model decreases with decreasing mesh size, further mesh refinement would produce even better agreement between the numerical results and experimental data.

In the last stage of these computations, the recently implemented implicit/explicit algorithm was used for the solution of the problem. Due to the oscillatory behavior of the solution (caused by the high speed flow) near the leading edge, the maximum bound for CFL number was set to a rather low value of 2. On the final adapted mesh this corresponds to only 6.6% of domain selected as implicit, with the implicit elements clustered in the near-wall region. Based on this zone selection, the cost reduction factor with respect to the fully implicit and fully explicit schemes was 0.06 and 0.65, respectively. This means that the implicit /explicit algorithm converged about 17 times faster than the fully implicit

algorithm (the only method available previously in the code) and about 2 times faster than the fully explicit algorithm, with an additional beneficial effect of smoothing the oscillatory tendencies of the solution near the plate tip.

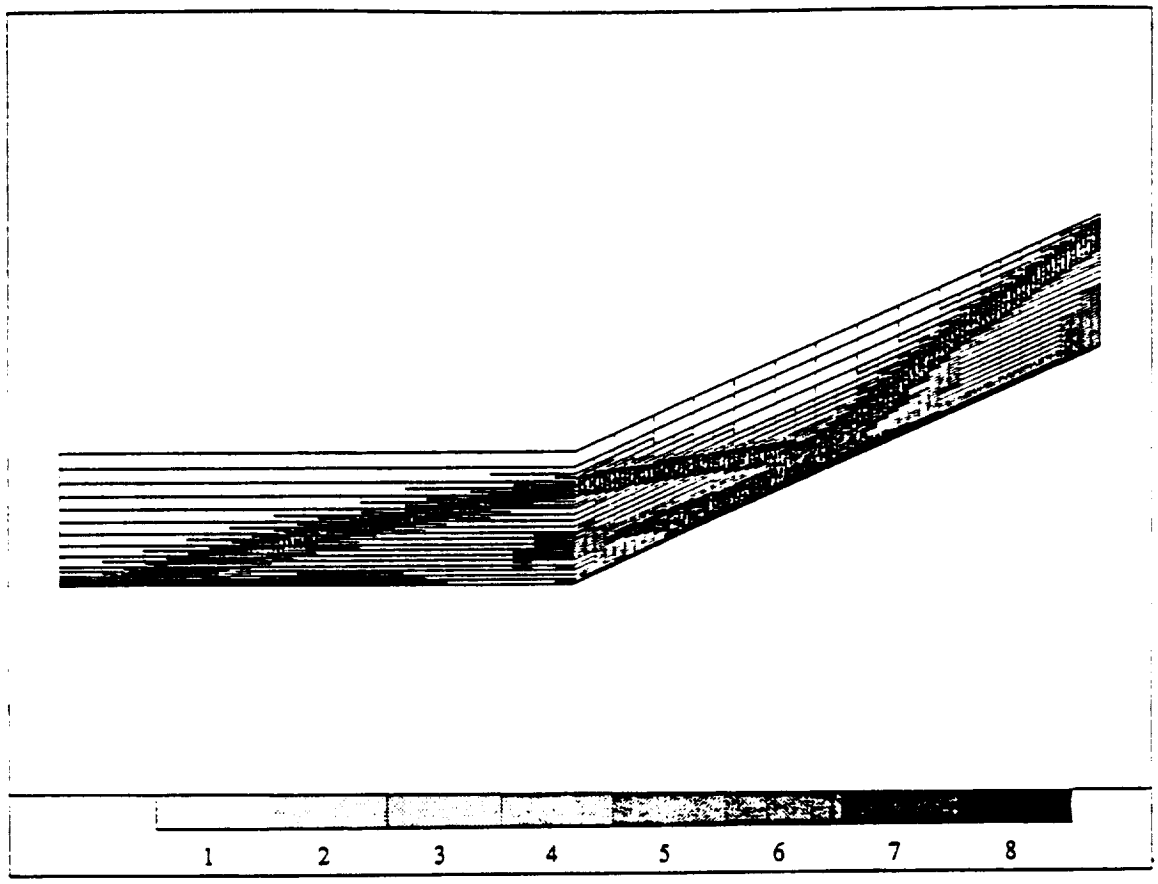


Figure 7.72: Holden problem, $Re = 72,000$, h-adapted mesh with three levels of refinement.

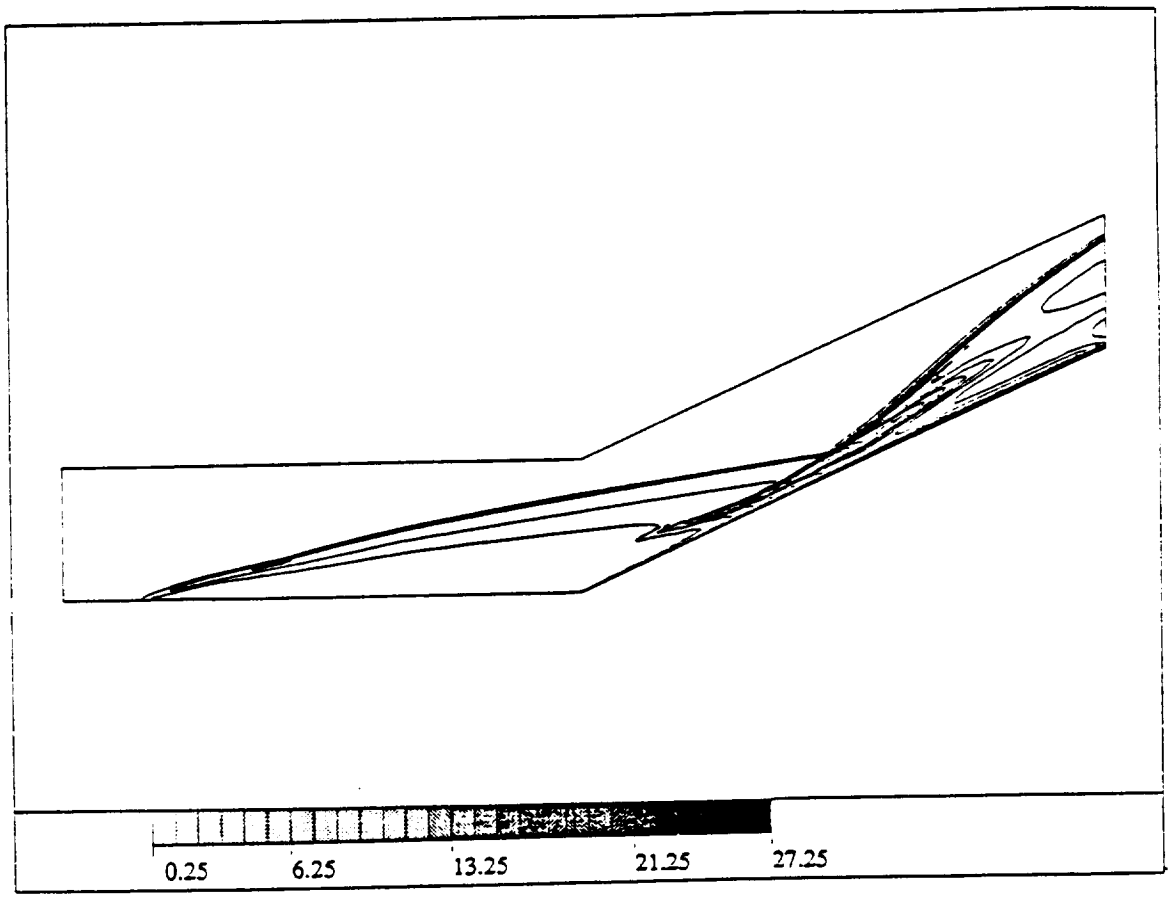


Figure 7.73: Holden problem, $Re = 72,000$, density contours for an h-adapted mesh.

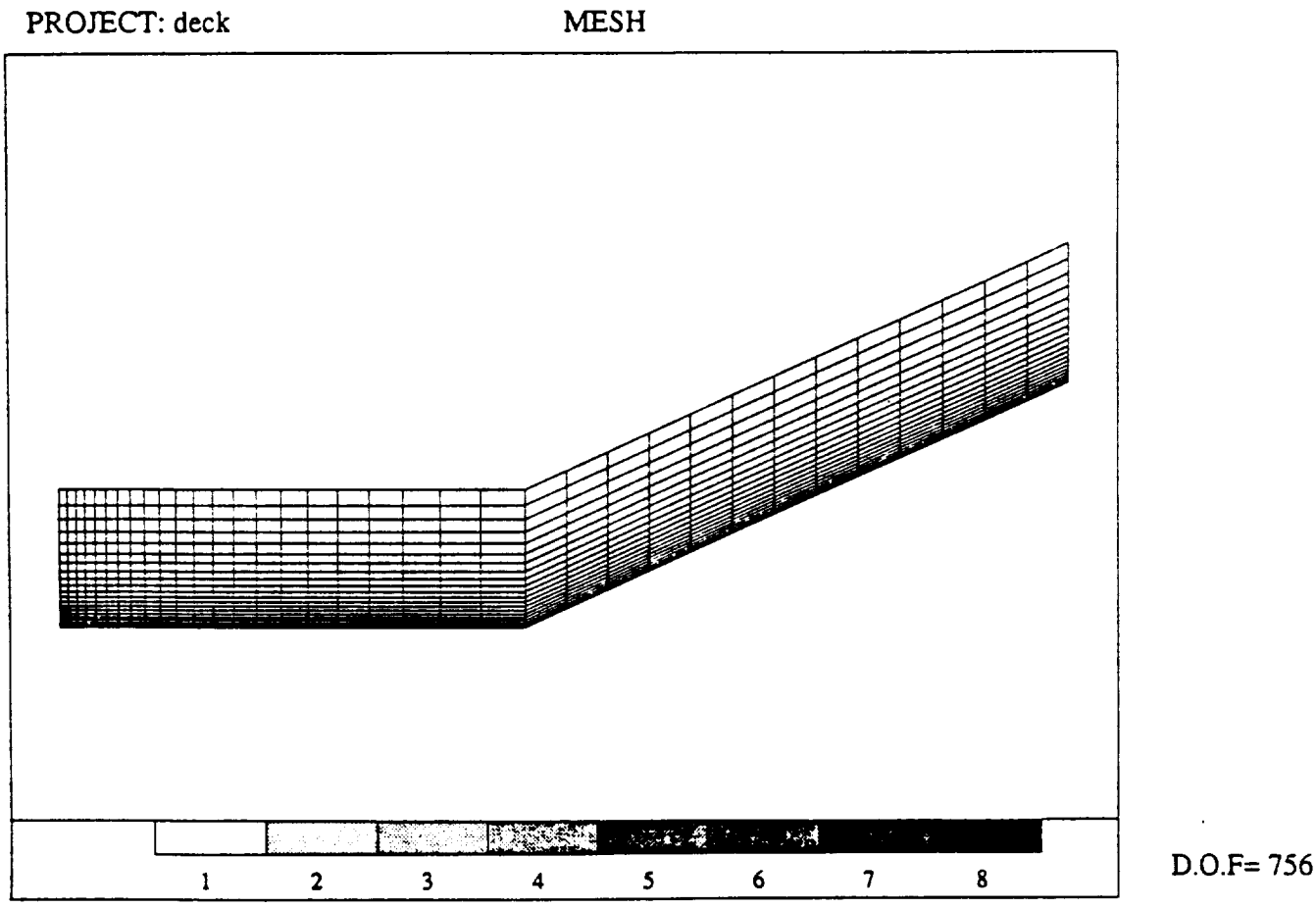


Figure 7.74: Holden problem, $Re = 72,000$, initial mesh used for the one–step method.

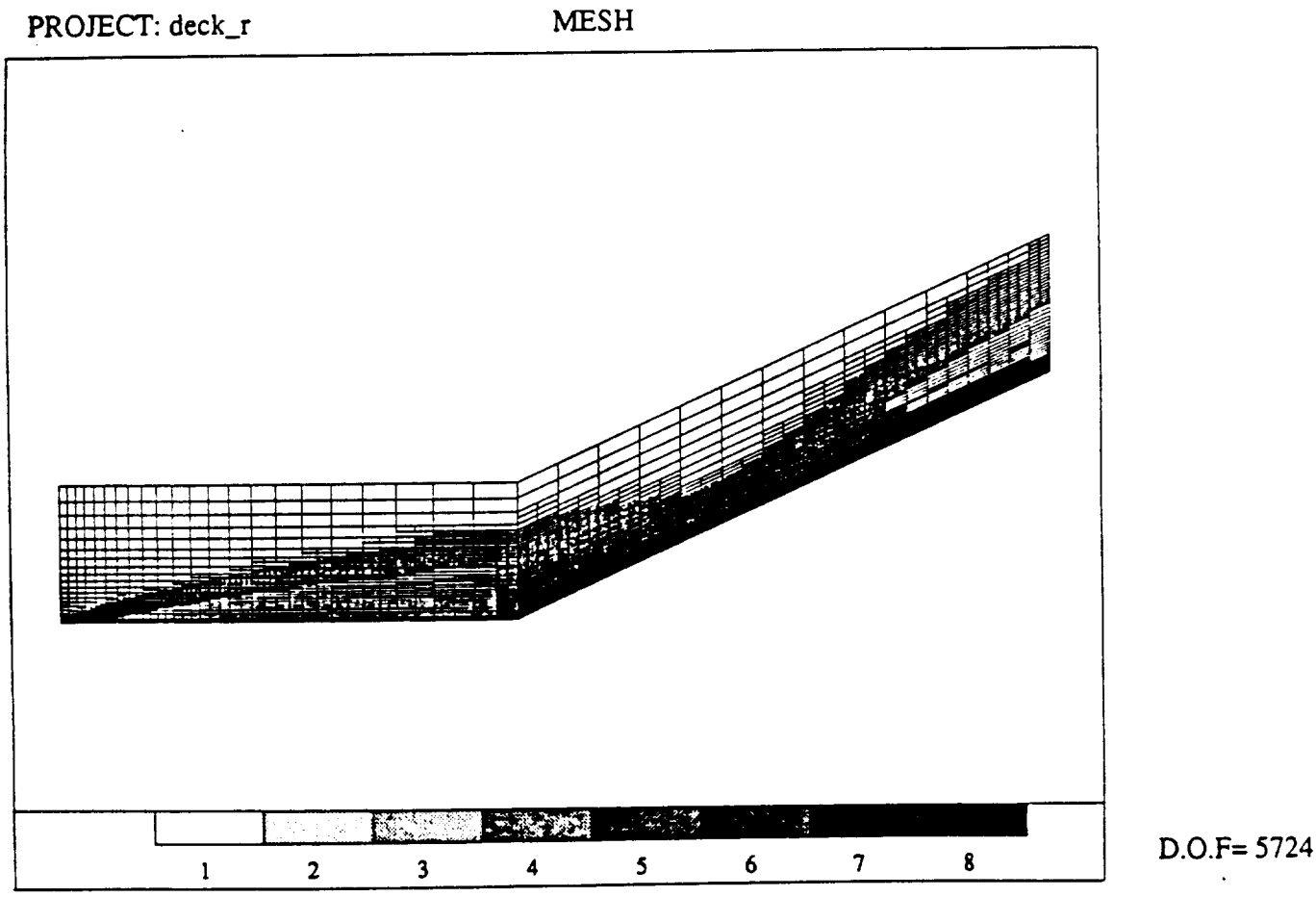


Figure 7.75: Holden problem, $Re = 72,000$, h-adapted mesh.

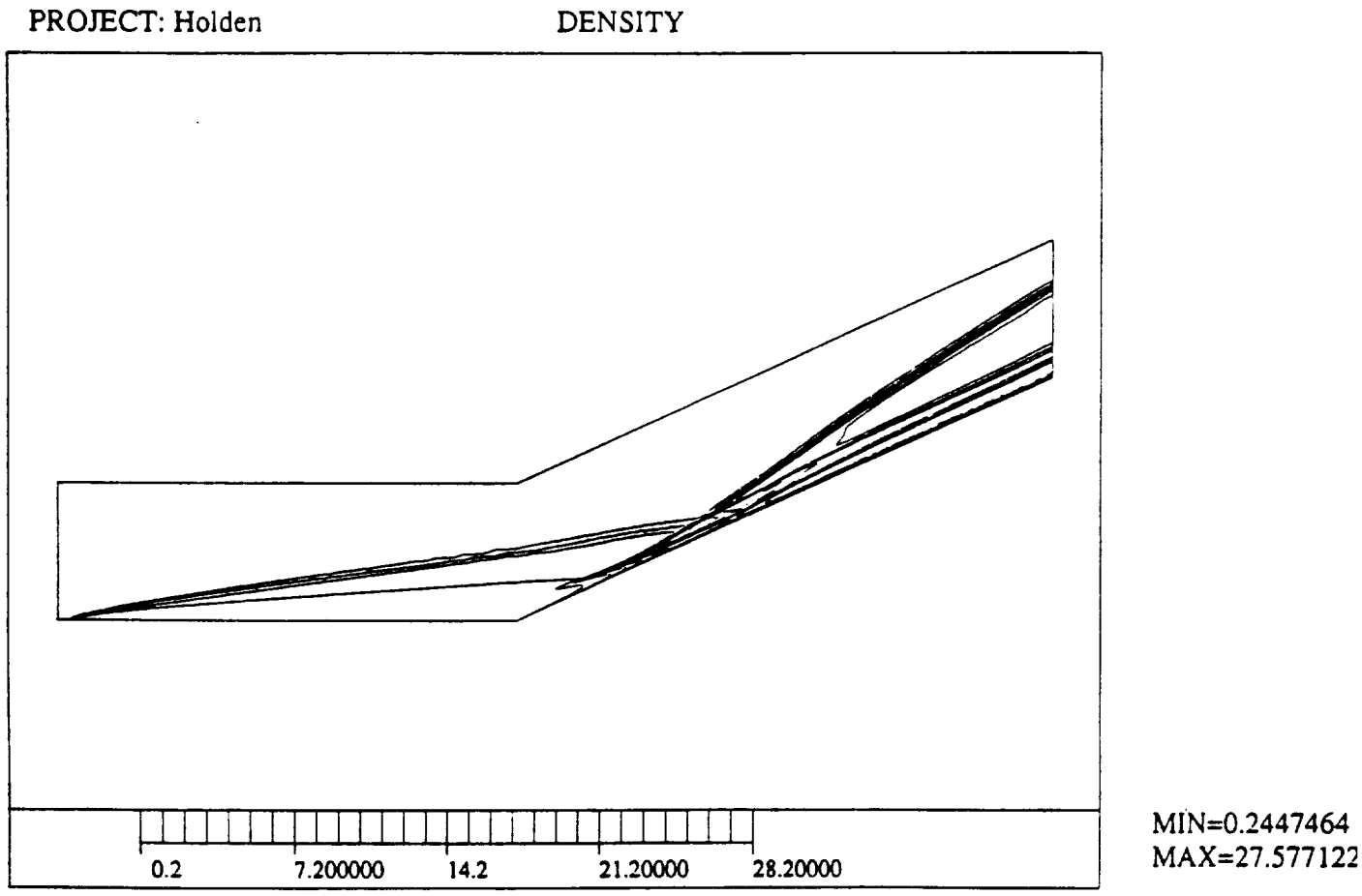


Figure 7.76: Holden problem, $Re = 72,000$, density contours for an h-adapted mesh.

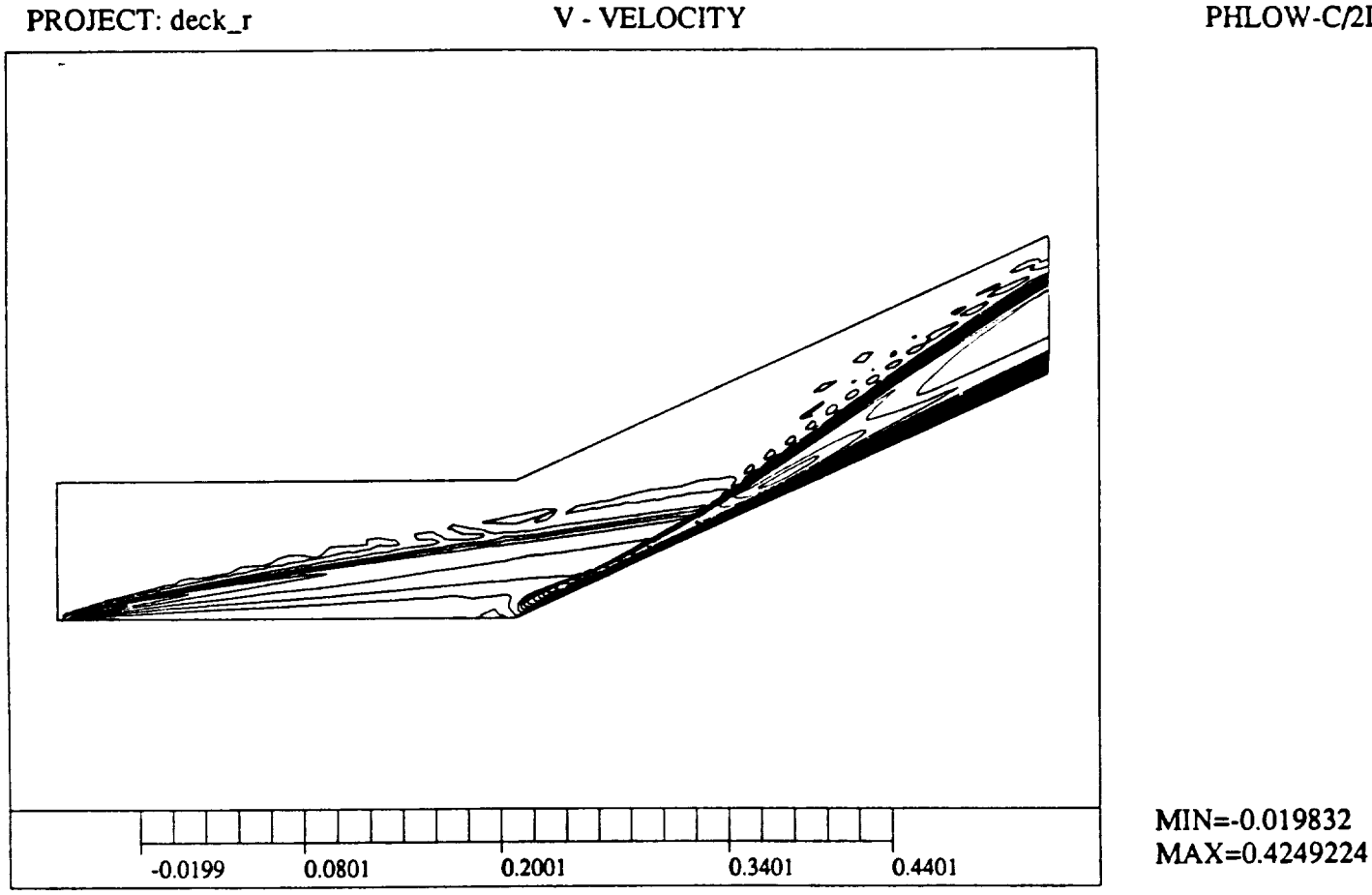


Figure 7.77: Holden problem, $Re = 72,000$, V-velocity contours for an h-adapted mesh.

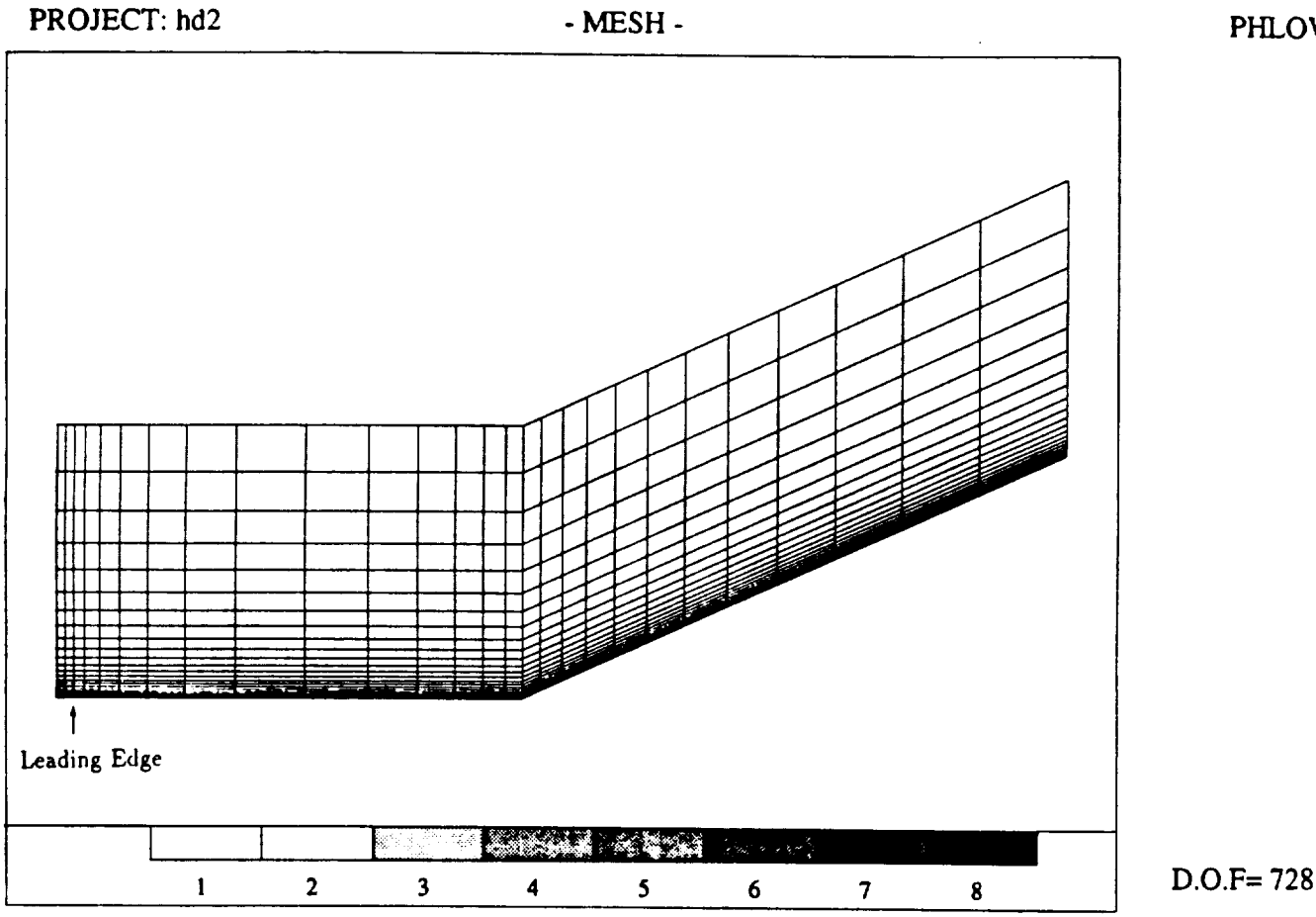


Figure 7.78: Initial mesh (referred as SM mesh).

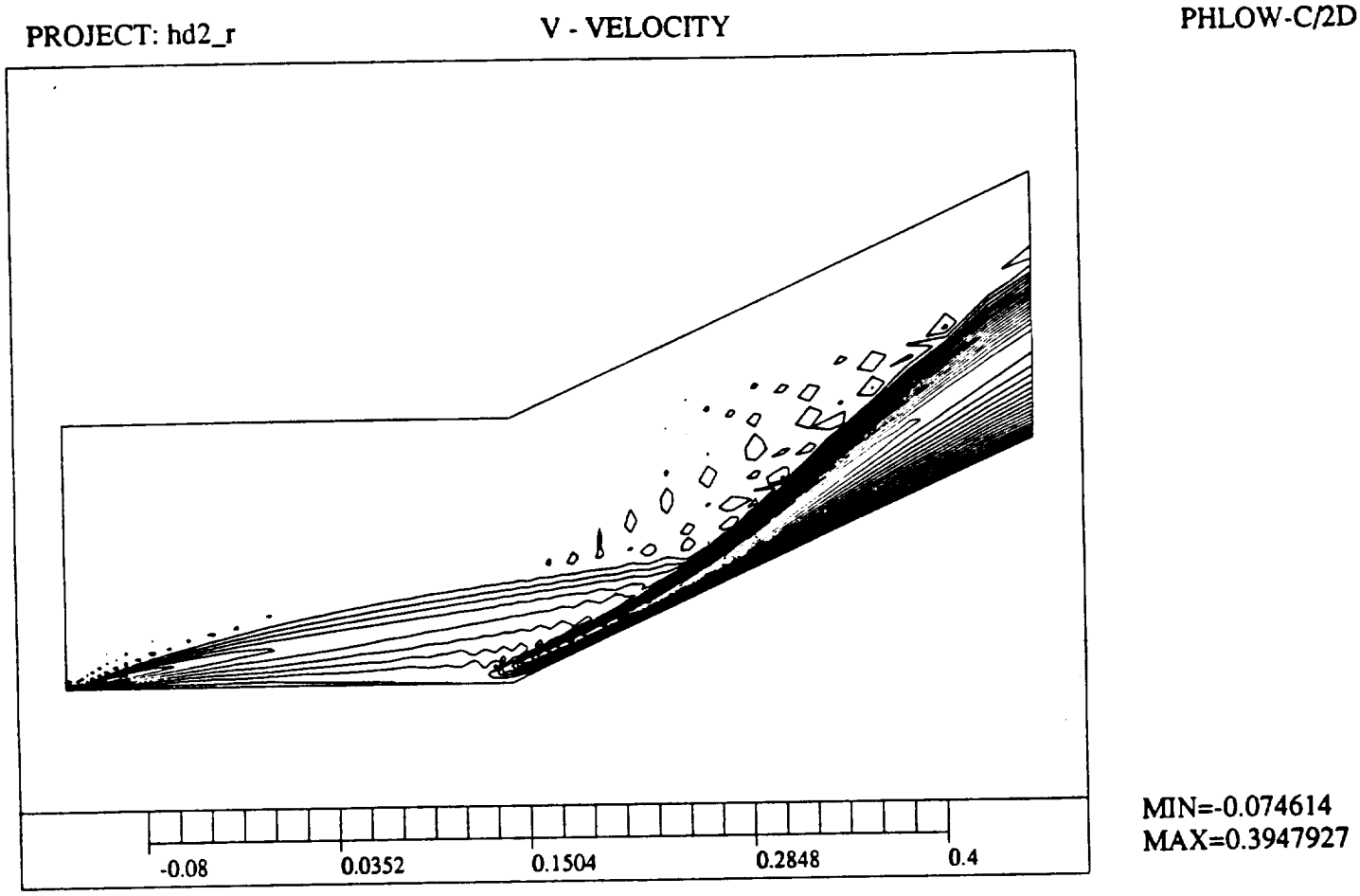


Figure 7.79: V-velocity contours for Lapidus AD model (on SM mesh with one level of uniform h-refinements.

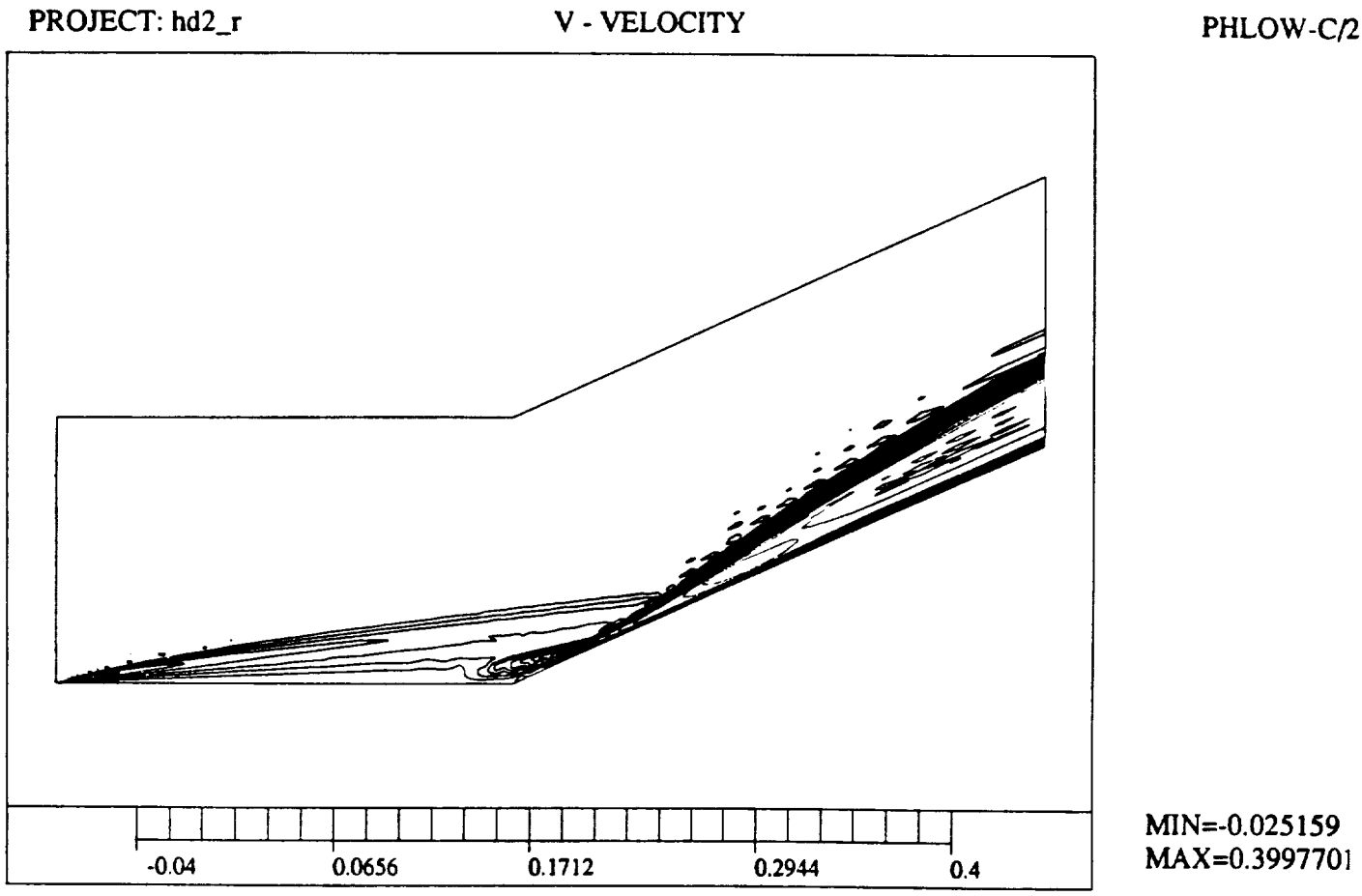


Figure 7.80: V-velocity contours for modified Morgan's AD model (on SM mesh with one level of uniform h-refinement.

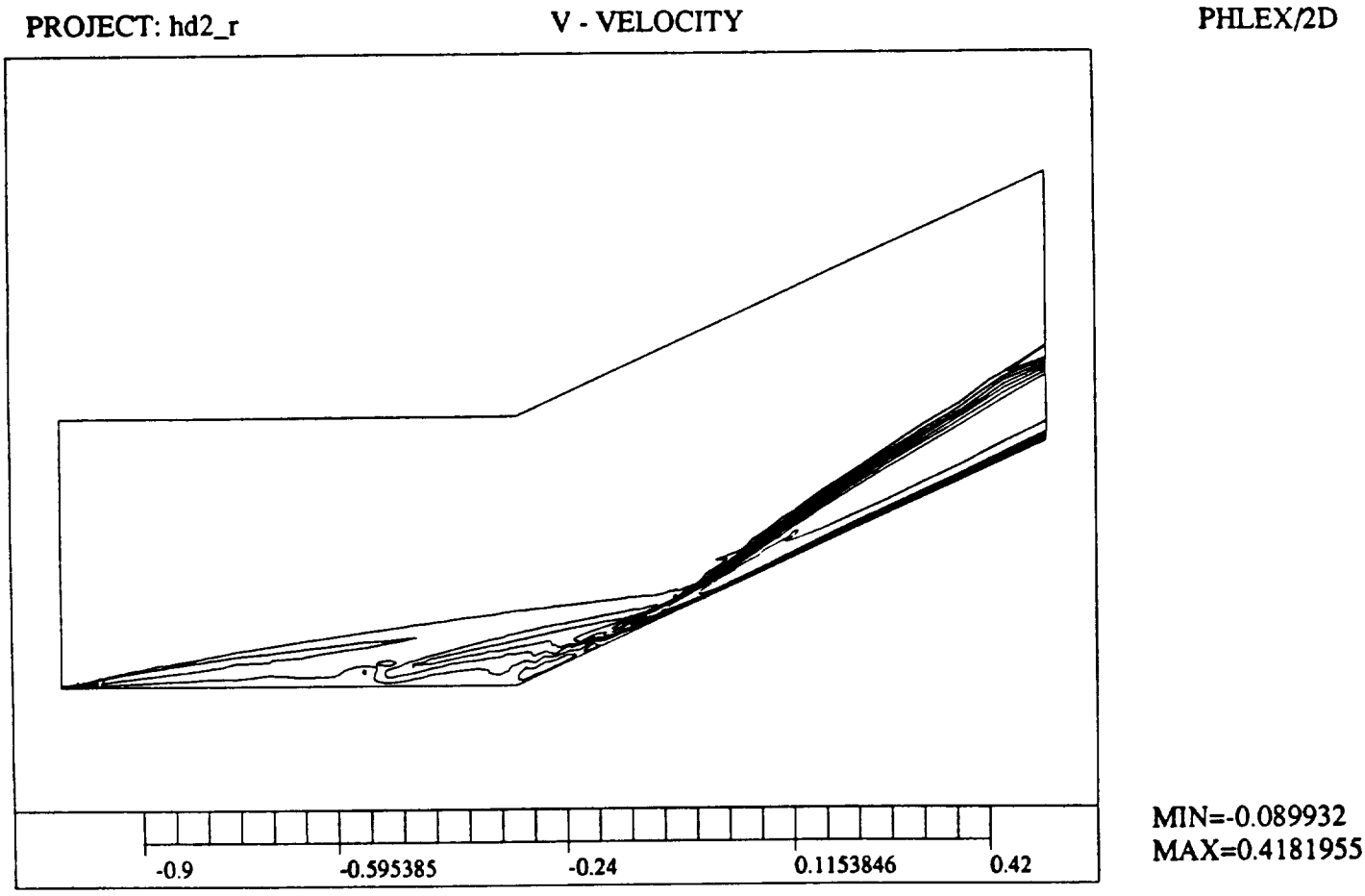


Figure 7.81: V-velocity contours for modified Morgan's AD model with fix (on SM mesh with one level of uniform h-refinement.

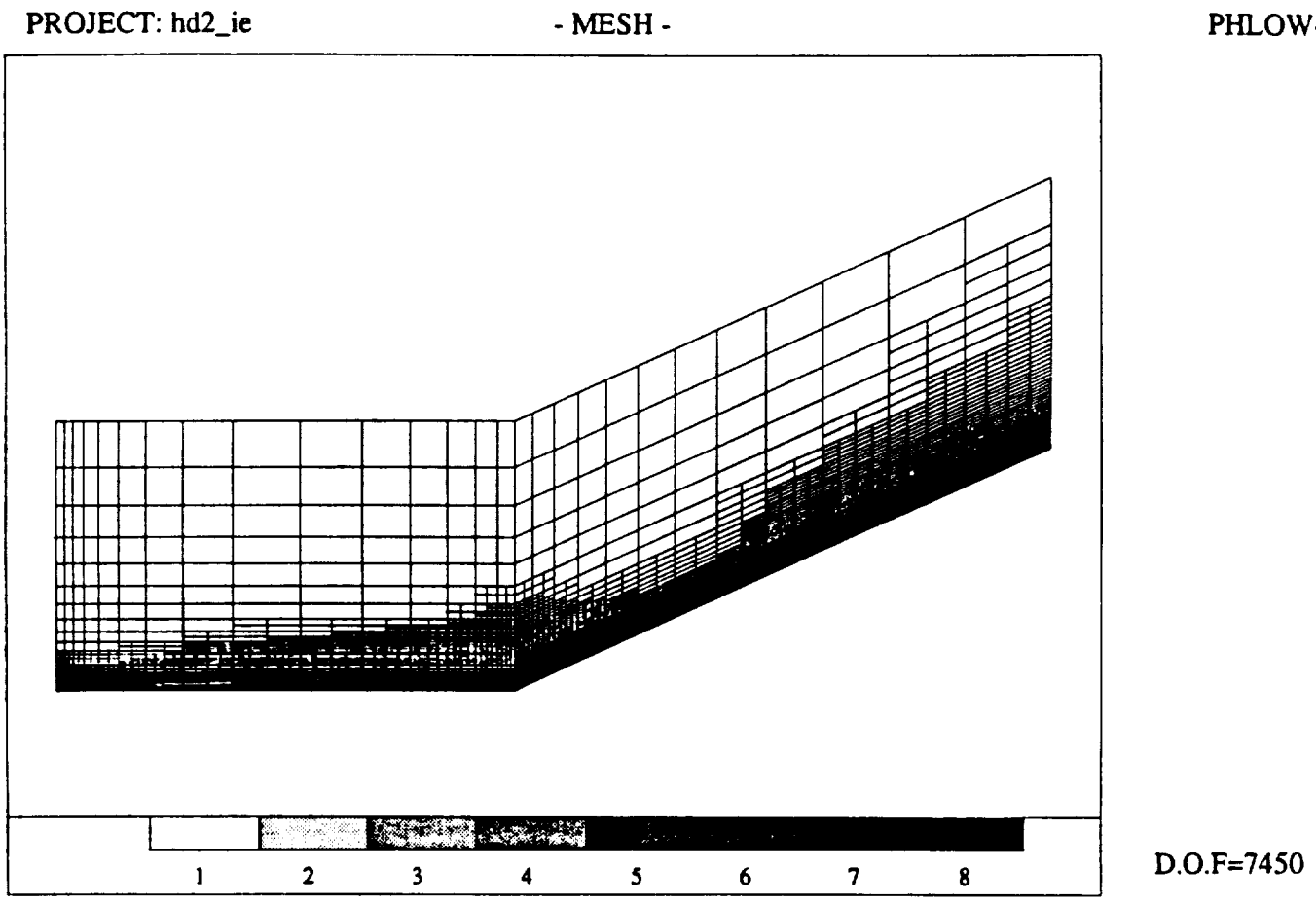


Figure 7.82: SM mesh after 2 levels of h-adaptation.

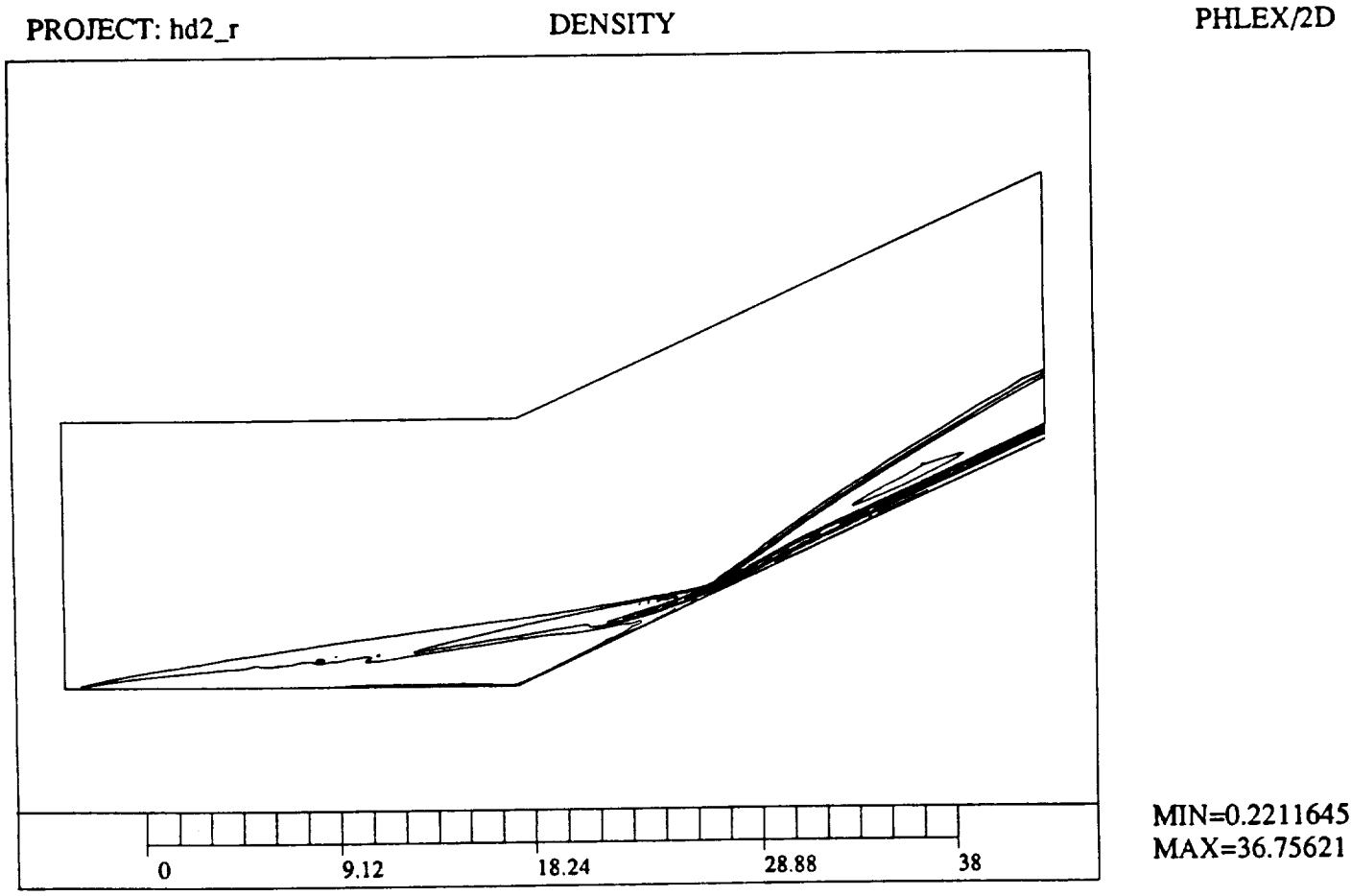


Figure 7.83: Density contours for an adapted SM mesh.

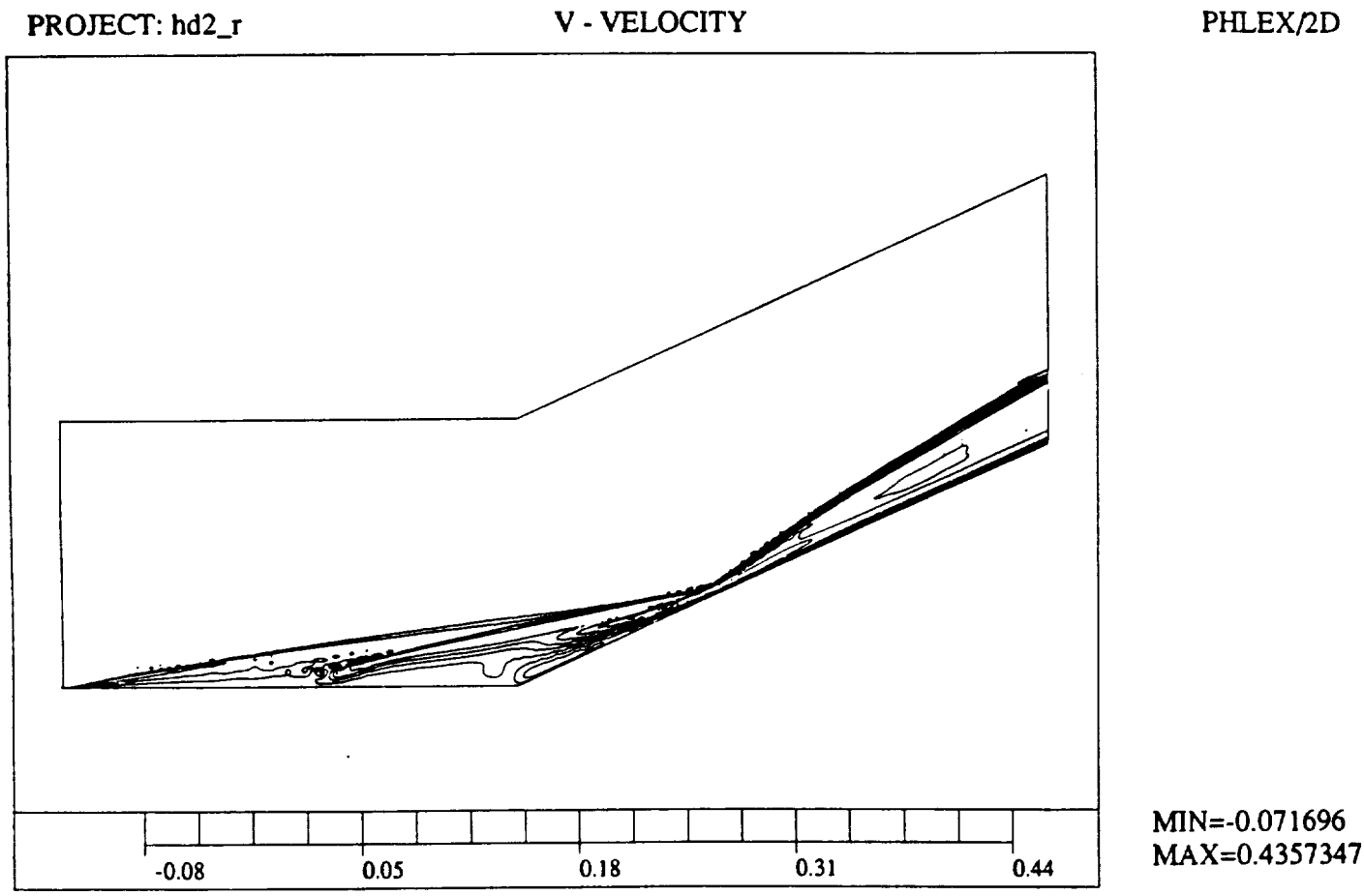


Figure 7.84: V-velocity contours for an adapted SM mesh.

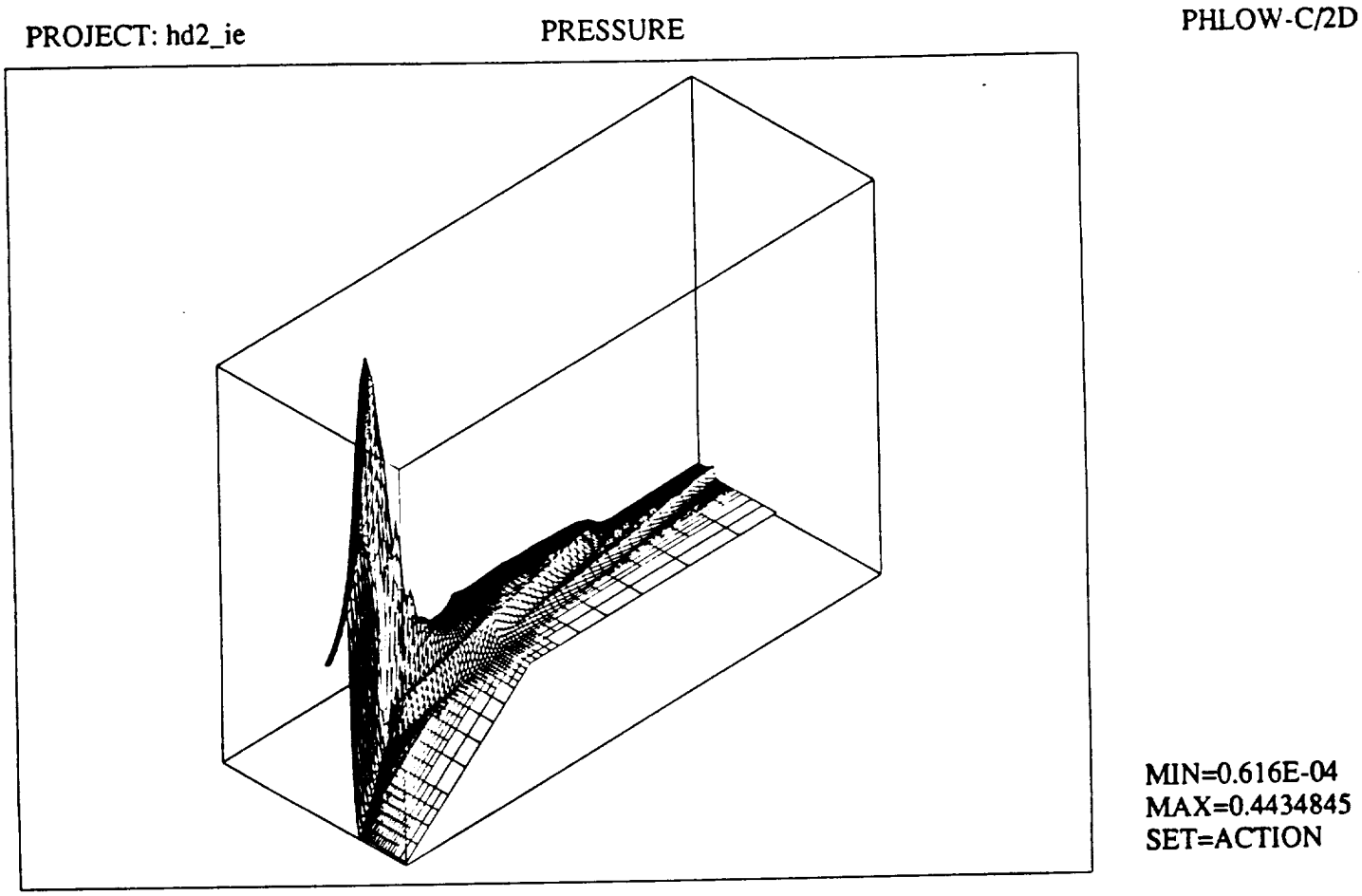


Figure 7.85: 3D view of pressure distribution for an adapted SM mesh (the protion of the mesh displayed covers the separation and reattachment regions.

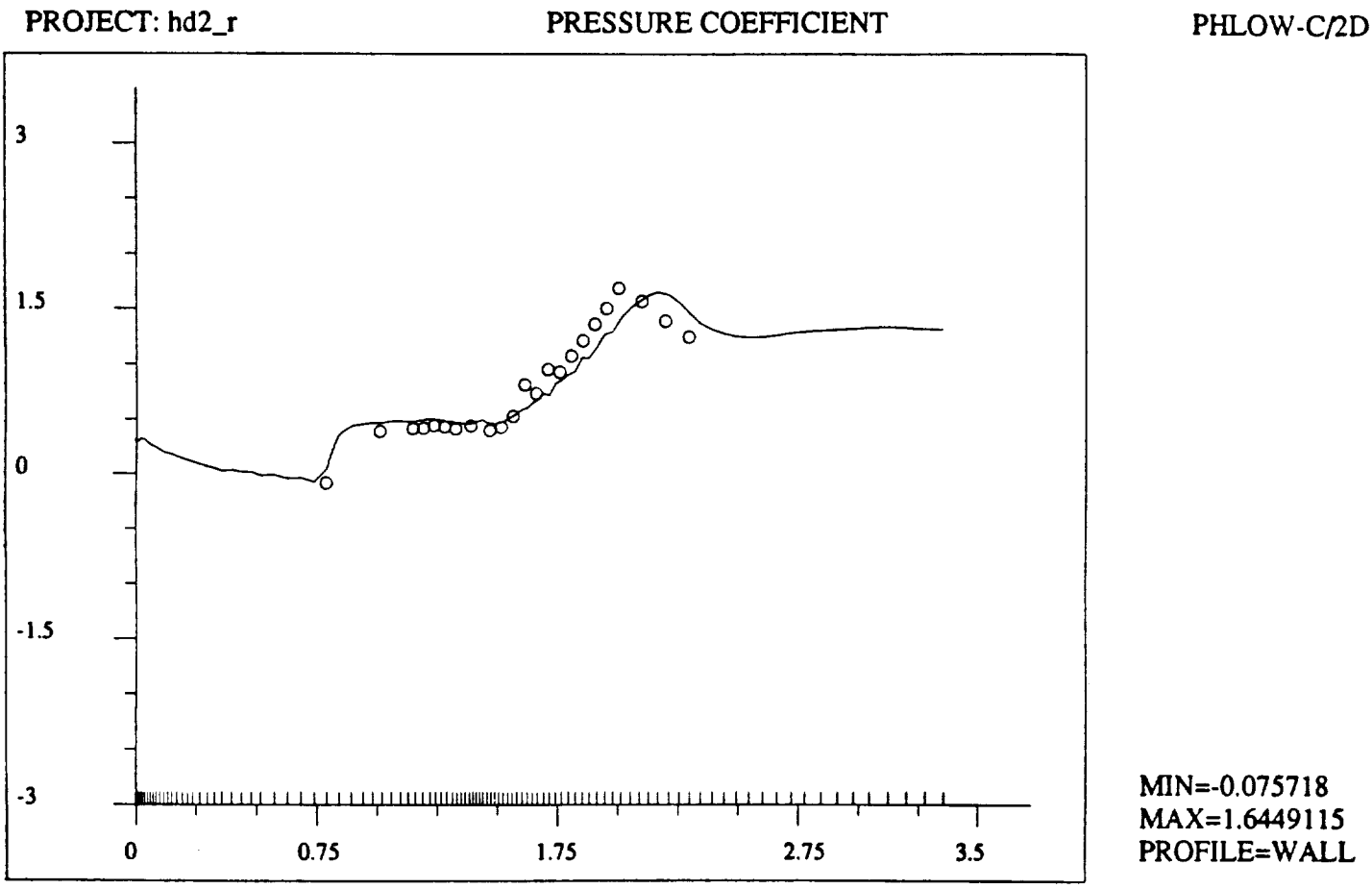


Figure 7.86: Pressure distribution along the plate for adapted SM mesh.

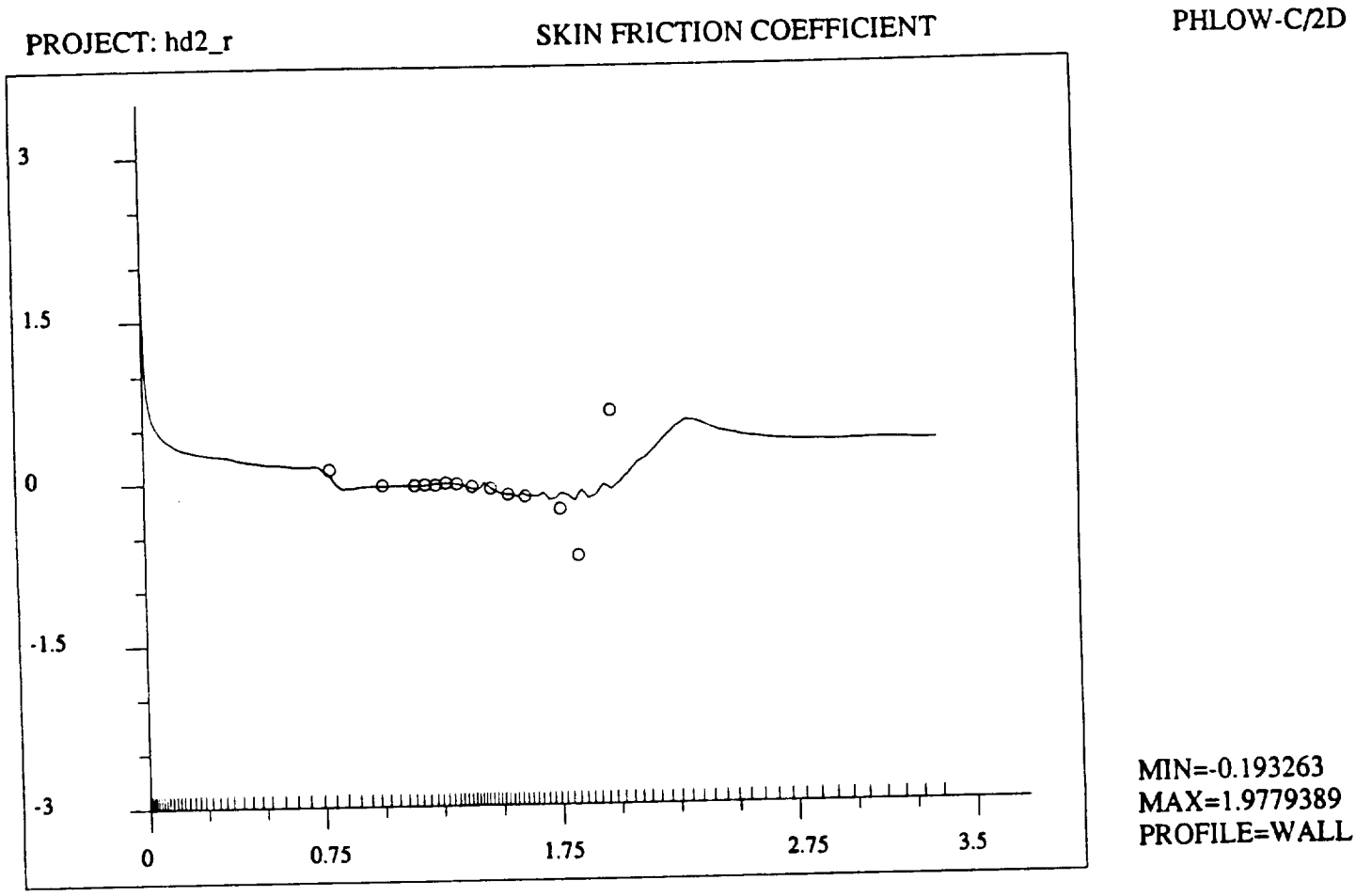


Figure 7.87: Skin friction distribution along the plate for adapted SM mesh.

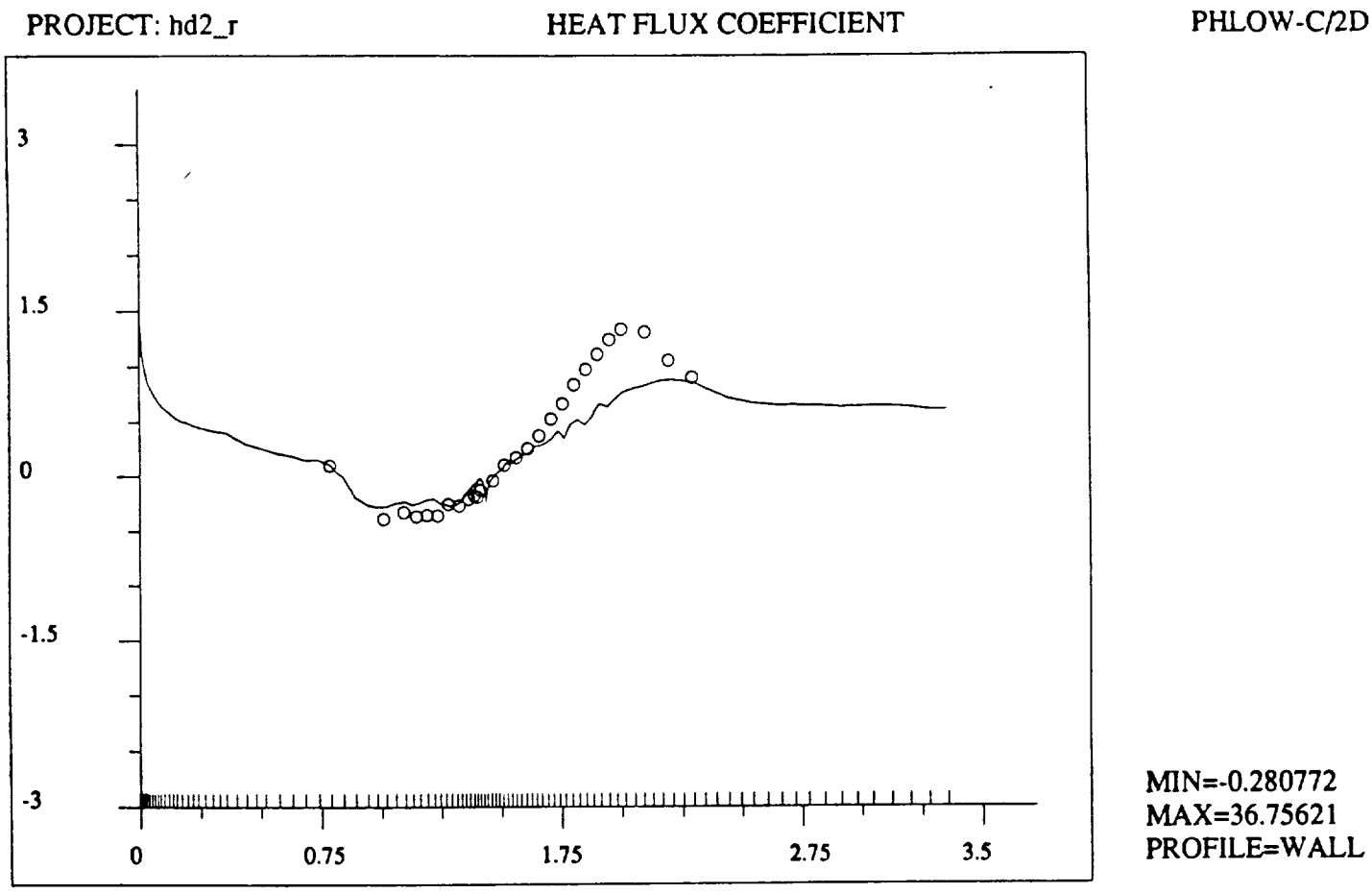


Figure 7.88: Heat flux distribution along the plate for adapted SM mesh.

Example 11: Rearward Facing Step with Strong Expansion

The third benchmark problem modeled herein is a supersonic, viscous, laminar separated flow over a rearward facing step. A schematic of the geometry is shown in Fig. 7.89 with several additional labels identifying regions of the domain which are referred to later. The experimental data provided by NASA Langley for this test case, as well as several other similar configurations, is found in Ref. [17]. In this reference, experiments were conducted for a range of test conditions: $T_\infty = 2700 \sim 5500°K, M_\infty = 3.95 \sim 4.27, Re_\infty/cm = 160 \sim 2200$. For the non-suction case, with no chemical reactions which is our focal point in this example, the experimental data includes a heat transfer distribution and a surface pressure distribution downstream of the step.

The experimental setup contained two sets of control data from which we selected the case with the maximum step height. For this case, $h = 1.02$ cm, and the following flow conditions exist:

$$\begin{aligned} M_\infty &= 4.08 \\ Re_{\infty,h} &= 1650 \\ T_\infty &= 3750°K \\ T_{\text{wall}} &= 297°K \end{aligned}$$

The initial mesh for this problem consists of 28 by 8 linear elements with some clustering near the wall as shown in Fig. 7.89. The initial conditions were defined as a uniform flow in the whole domain, and the problem was first solved with 2 levels of uniform refinements which corresponds to 3729 degrees of freedom. A very small time step was initially required to avoid the negative densities and pressures near the corner of the backstep during start up from free stream conditions. After the flow stabilized, we applied the h-adaptation option to begin resolving the shocks and boundary layer regions. The flow features of specific interest in this problem required high resolution of the flow field variables in the regions of the boundary layer along the wall, and around the two corners of the backstep.

The corresponding mesh with 3 levels of h-refinement is shown in Fig. 7.90, and a 3D view of the density is presented in Fig. 7.91. Large gradients of the solution occur near the wall both upstream and downstream of the backstep. For the upstream portion of the computational domain there are approximately 5 layers of linear elements in the boundary layer which are probably adequate to roughly resolve the flow features therein. In the downstream portion of the computational mesh, however, there are only approximately two layers of linear elements which are undoubtedly insufficient to capture the heating rates and other fine scale phenomena.

To adequately resolve the downstream portion of the mesh we applied a uniform h-

refinement to four layres of elements near the wall in this region to arrive at the mesh (referred to as M3-mesh) as shown in Fig. 7.92. On this mesh we continued the solution process with a time-step size corresponding approximately to $CFL = 5$. The resulting pressure contours on the entire mesh and the temperature contours in the backstep and downstream regions are shown shown in Figs. 7.93 and 7.94, respectively. From these plots one can observe the overall patterns of the leading edge shock, the expansion fan at the corner of the step, the reattachment shock downstream of the step, and the boundary layers near the wall. The density and heat flux coefficient along the wall downstream of the backstep are also profiled in Figs. 7.95 and 7.96, respectively.

From this point we continued the solution process along two different but parallel paths to investigate the effectiveness of h-adaptation versus p-enrichment procedures. The first approach consisted of applying the h-refinement/unrefinement algorithm to the M3-mesh using the element residual error estimate scheme presented in Sec. 4.1 as the adaptation driver. This adaptive pass resulted in the refinement of several elements in the following regions: the leading edge, the boundary layers near the wall upstream and downstream of the step, the corner of the step, and the reattachment shock zone. The resulting h-adapted mesh (referred to as M4H-mesh) which consists of 9700 linear elements and 9633 degrees of freedom is shown in Fig. 7.97. After continuing to march the solution for approximately 6 seconds of real time, the solution reached steady-state on the new mesh. The corresponding pressure contours, temperature contours, density profile, and heat flux coefficient are shown in Figs. 7.98 to 7.101. As expected, the enhanced mesh provides a better resolution of the flow variables in the reattachment shock and boundary layer regions than those obtained on the M3-mesh.

The second approach consisted of using the p-enrichment procedure proposed in Sec. 4.2 to adapt the mesh. Based on the solution obtained on the M3-mesh, the mesh was first adaptively h-unrefined (as in the M4H-mesh) to remove excess degrees of freedom. Then, for the elements with errors larger than the user specified threshold value (the same as the one used for the M4H-mesh), we applied the adaptive directional p-enrichment procedure. The directional adaptation indicators were calculated based on the derivatives of the solution, Eq. (4.42), and the values of b_1 and b_2 for selecting the enrichment directions were set to 0.3 and 0.7, respectively. The enriched mesh (referred to as M4P-mesh) is shown in Fig. 7.102. Figs. 7.103 and 7.104 show blowups of the mesh around the corner of the step and along the wall downstream of the step, respectively. Notice that the M4P-mesh consists of only 7774 degrees of freedom, which is almost 2000 less than the M4H-mesh. After marching the solution for 6 seconds of real time on this mesh, the corresponding pressure contours, temperature contours, density profile, and heat flux coefficient were extracted as shown in Figs. 7.105 to 7.108.

Qualitatively, the p-enriched M4P-mesh showed a similar overall improvement in the

solution over M3-mesh as the h-refined M4H-mesh did. The density profiles along the wall downstream the step of M4H-mesh and M4P-mesh appear to be virtually the same. The one significant difference in these two results appears in the prediction of heat flux coefficients which shows about a 20 to 30 percent variation. Since both cases predicted the same location of maximum heat transfer, we believe that the descripency of wall heat transfer rate in the h-adapted mesh is caused by the still insufficient mesh clustering in the direction normal to the boundary layer near the wall. Based on our experience in solving the other two benchmark problems (the Holden's problem and the blunt body with impinging shock problem), where the heat transfer rates were always underpredicted on h-meshes, we tend to conclude that the p-enriched mesh in the boundary layer region is providing a more effective use of degrees of freedom than h-refined mesh.

It should be noted that for this benchmark problem the experimental data presented in Ref. [17] do not provide enough detailed information to compare our numerical results with. Furthermore, unlike the other reference (e.g. [19]) where real gas effects were taken into account, our assumption of perfect gas at $\gamma = 1.4$ may also make the comparison between numerical and experiment data difficult at this time. Note thar the real gas effect will be considered in the next year of this project.

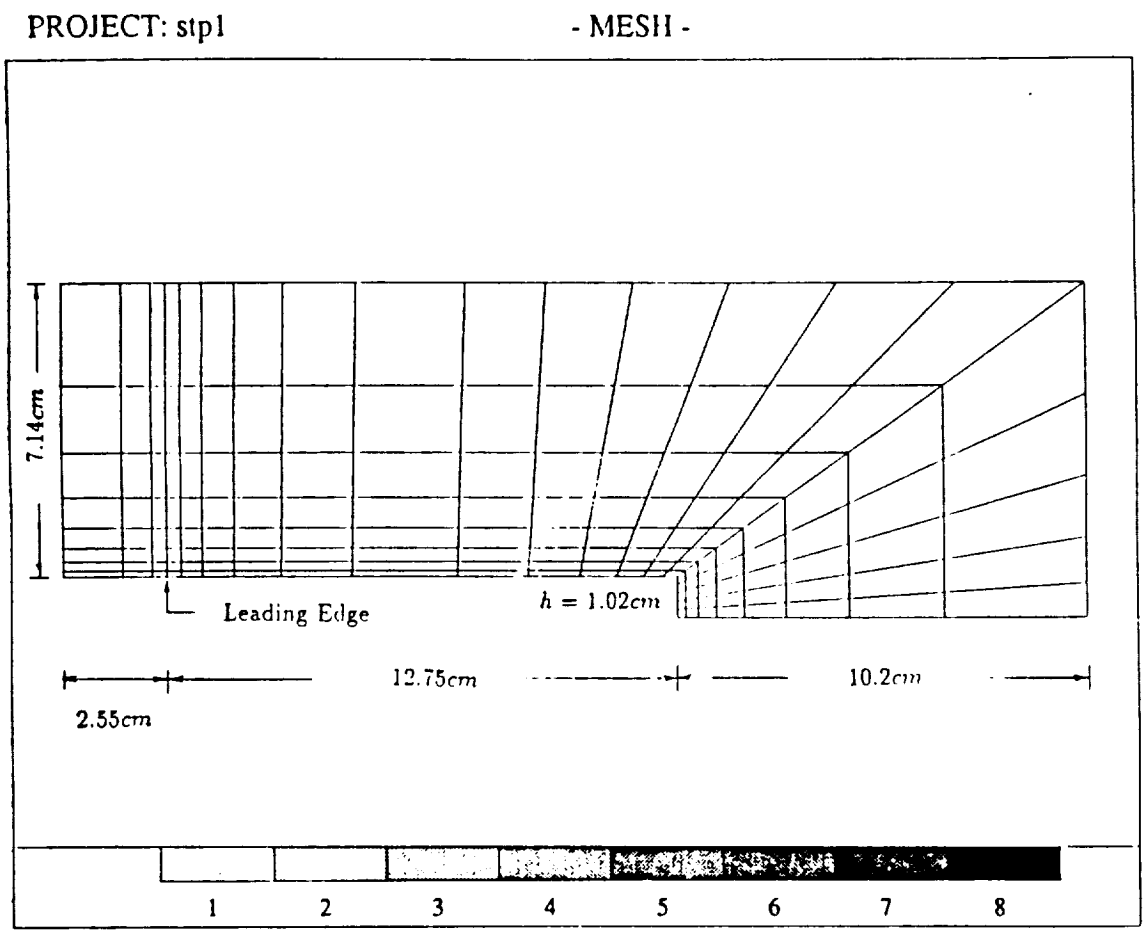


Figure 7.89: Rearward facing step and initial mesh.

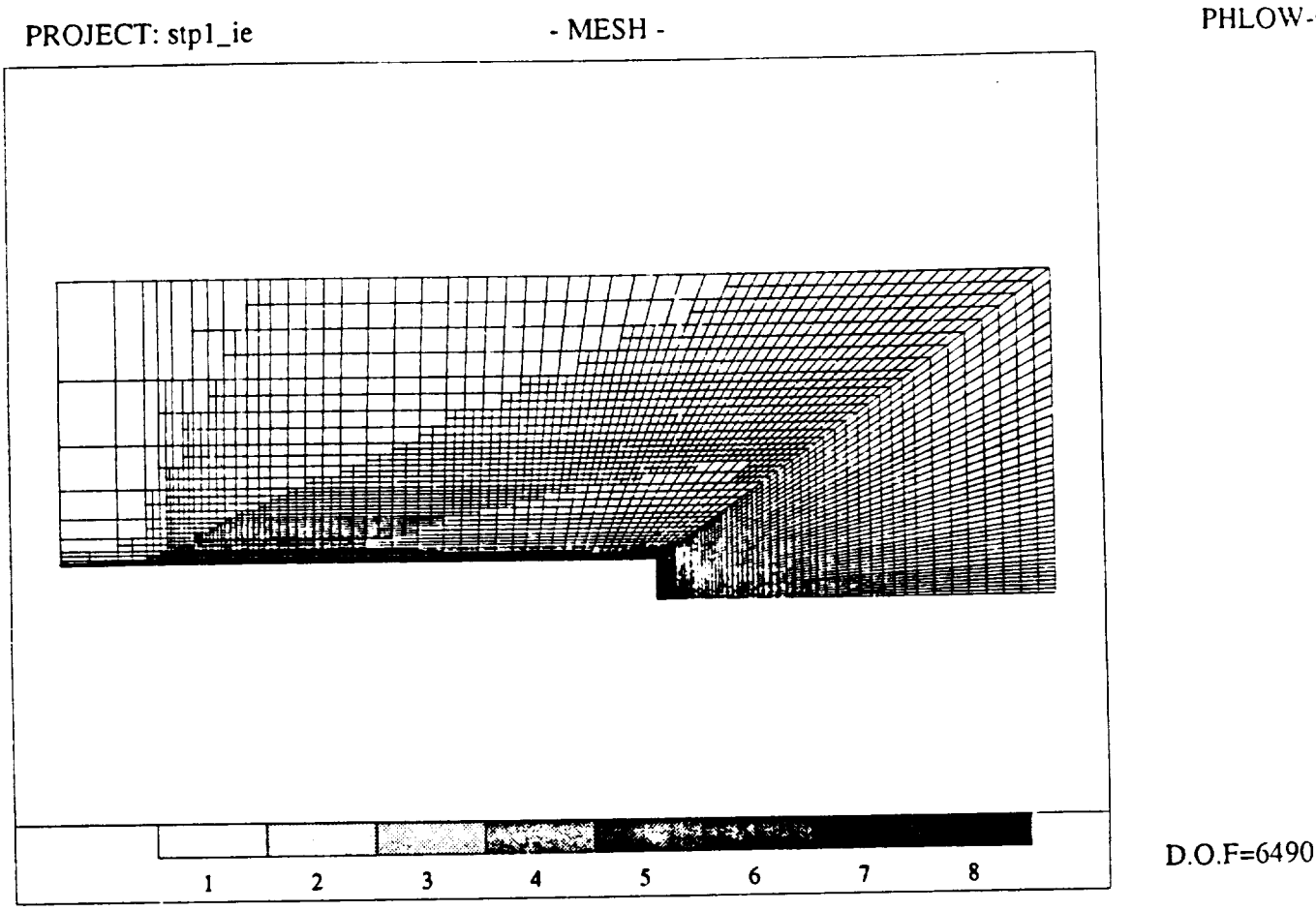


Figure 7.90: Adapted mesh with 3 levels of h-refinement.

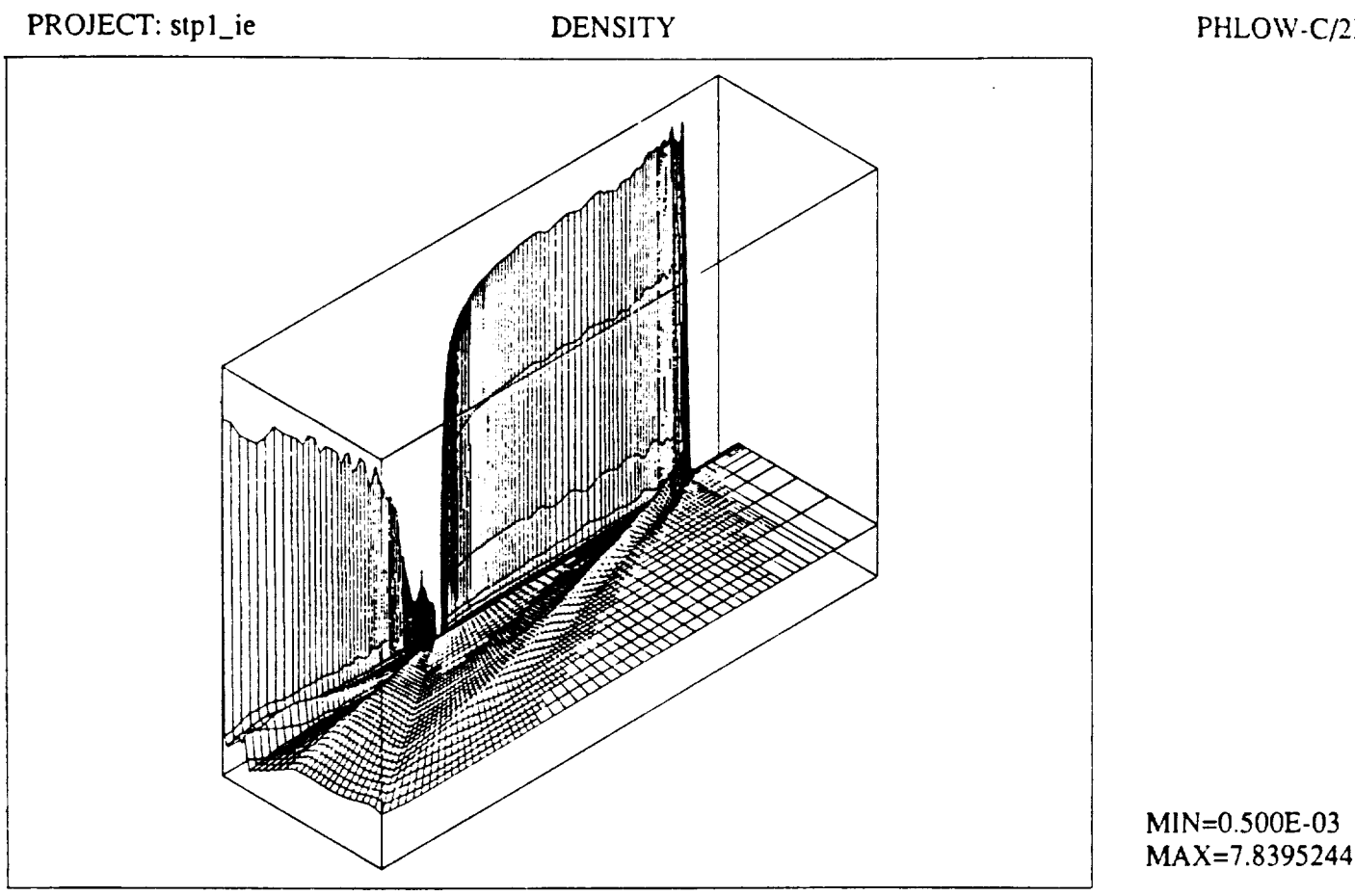


Figure 7.91: 3D view of density on adapted mesh.

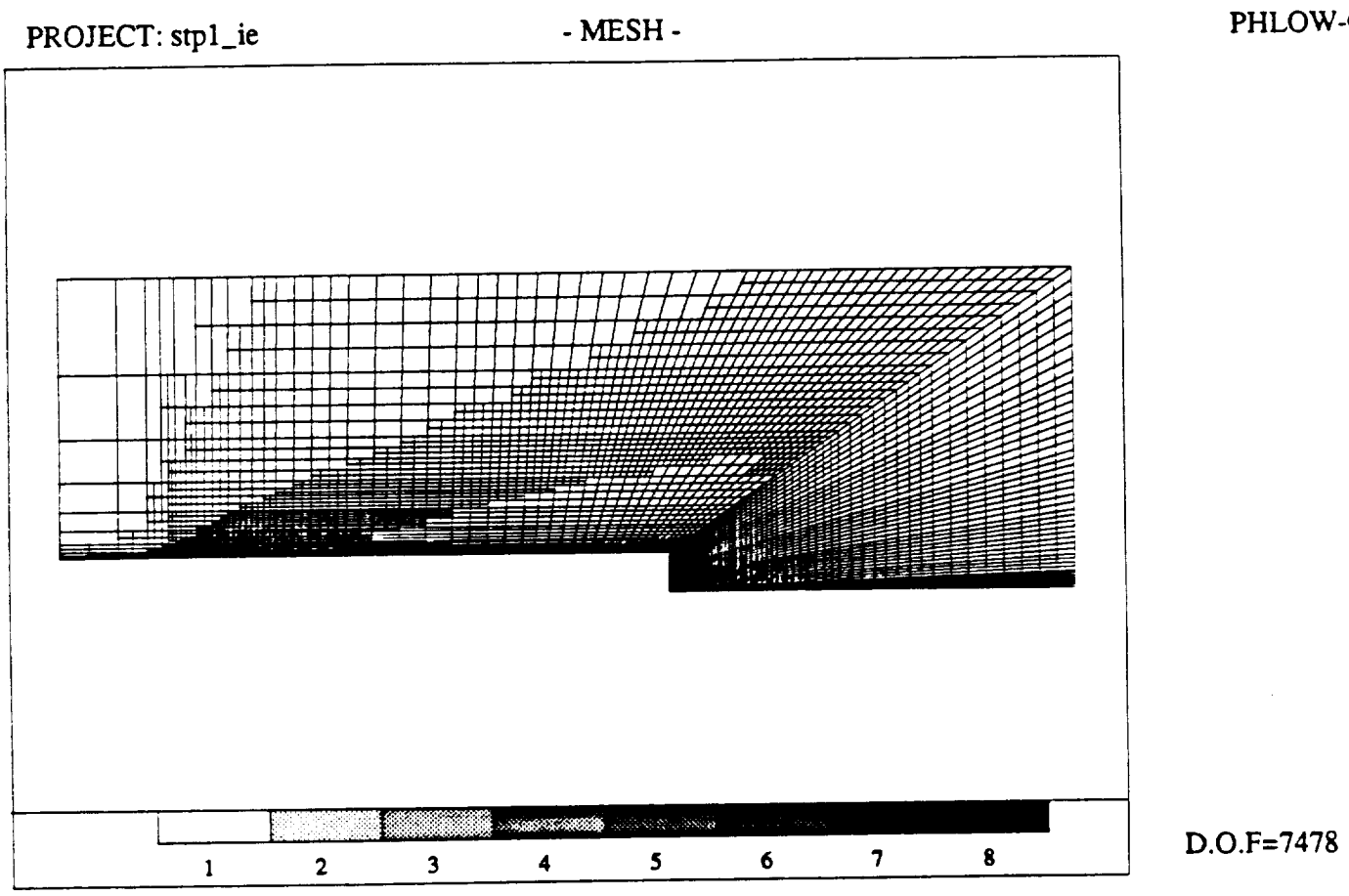


Figure 7.92: Adapted mesh with uniform h-refinement near the wall downstream of the backstep (M3-mesh).

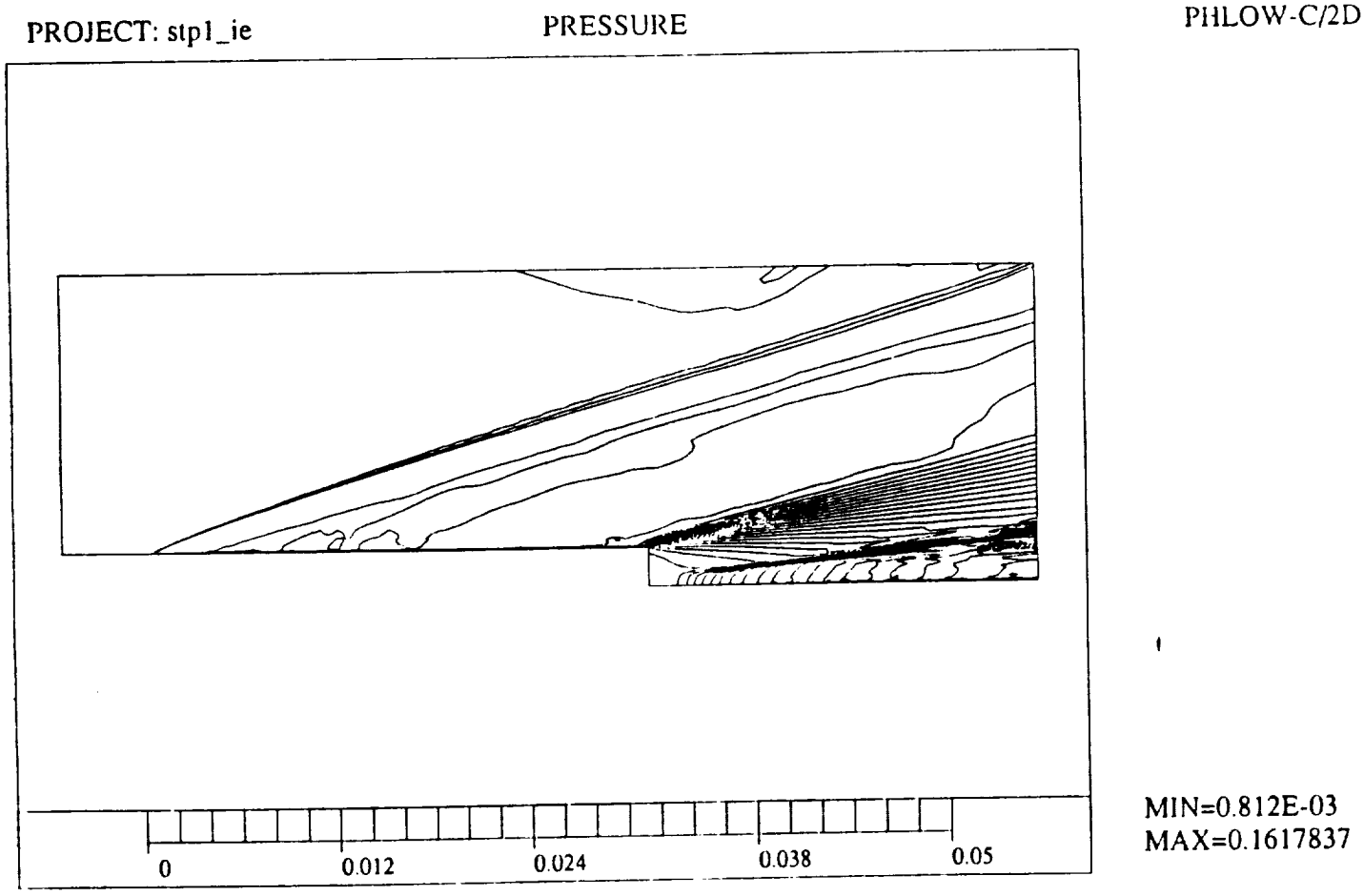


Figure 7.93: Pressure contours, M3-mesh

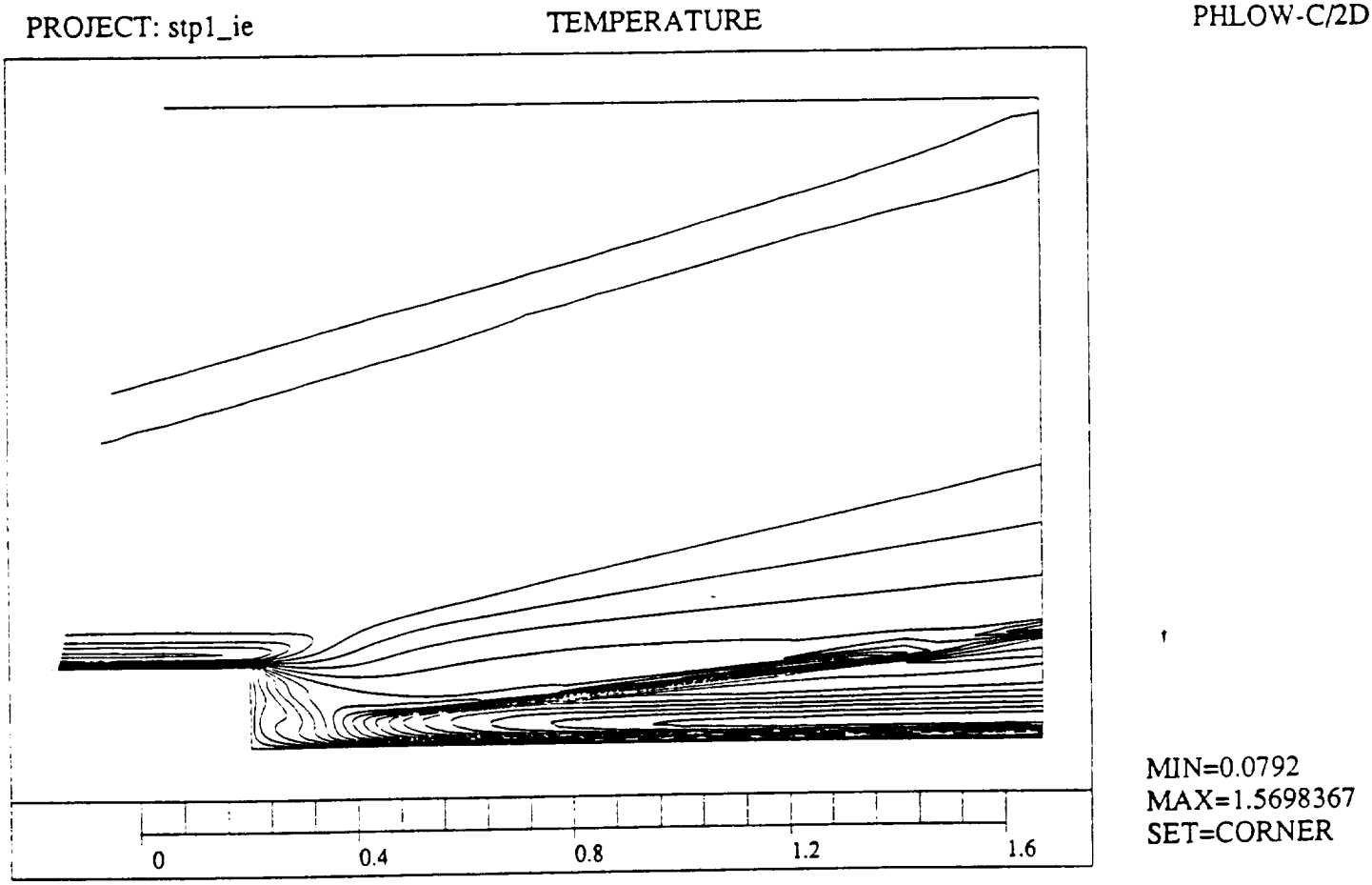


Figure 7.94: Temperature contours in the backstep region, M3-mesh.

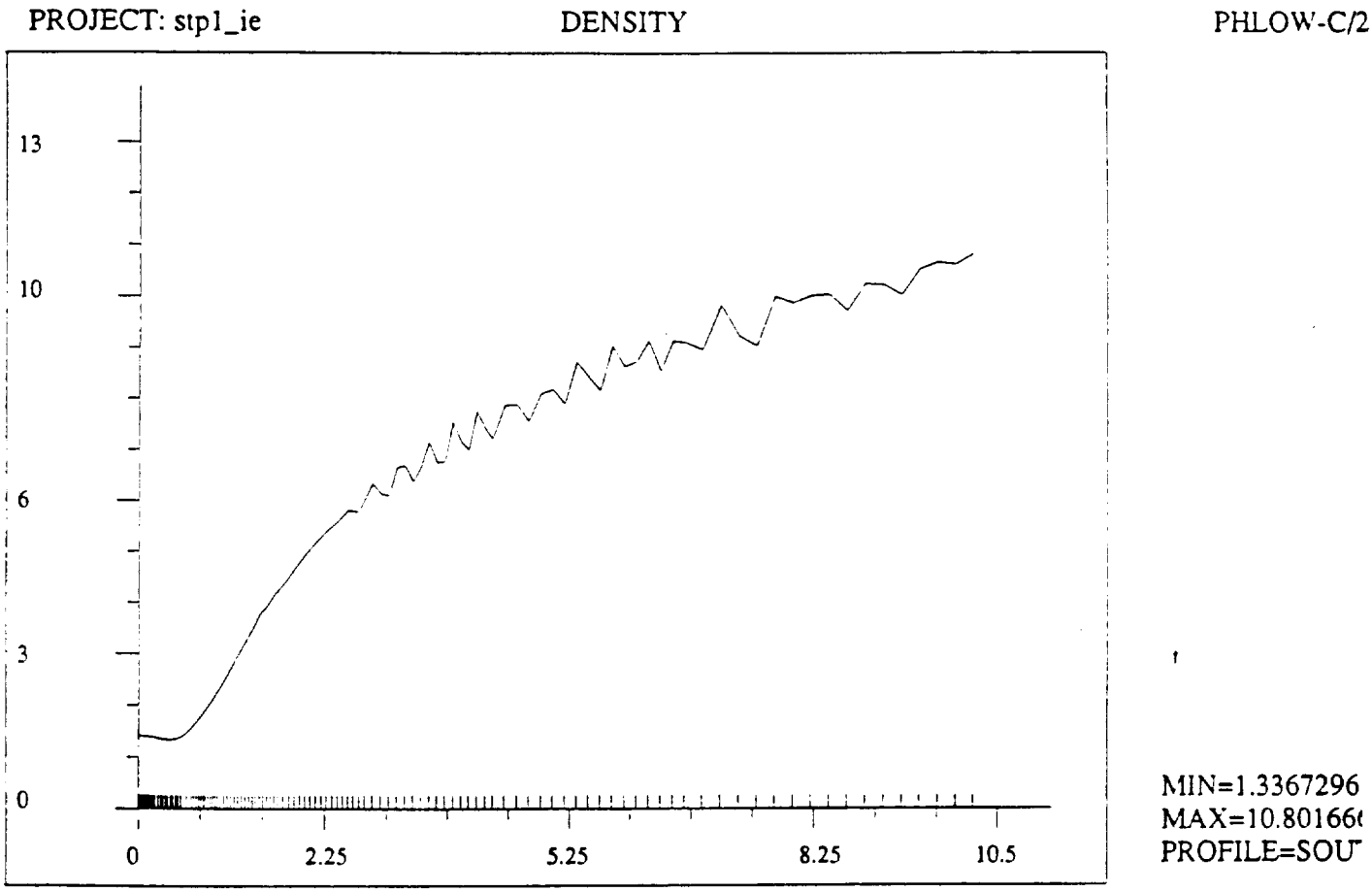


Figure 7.95: Density profile along the wall downstream of the backstep, M3-mesh.

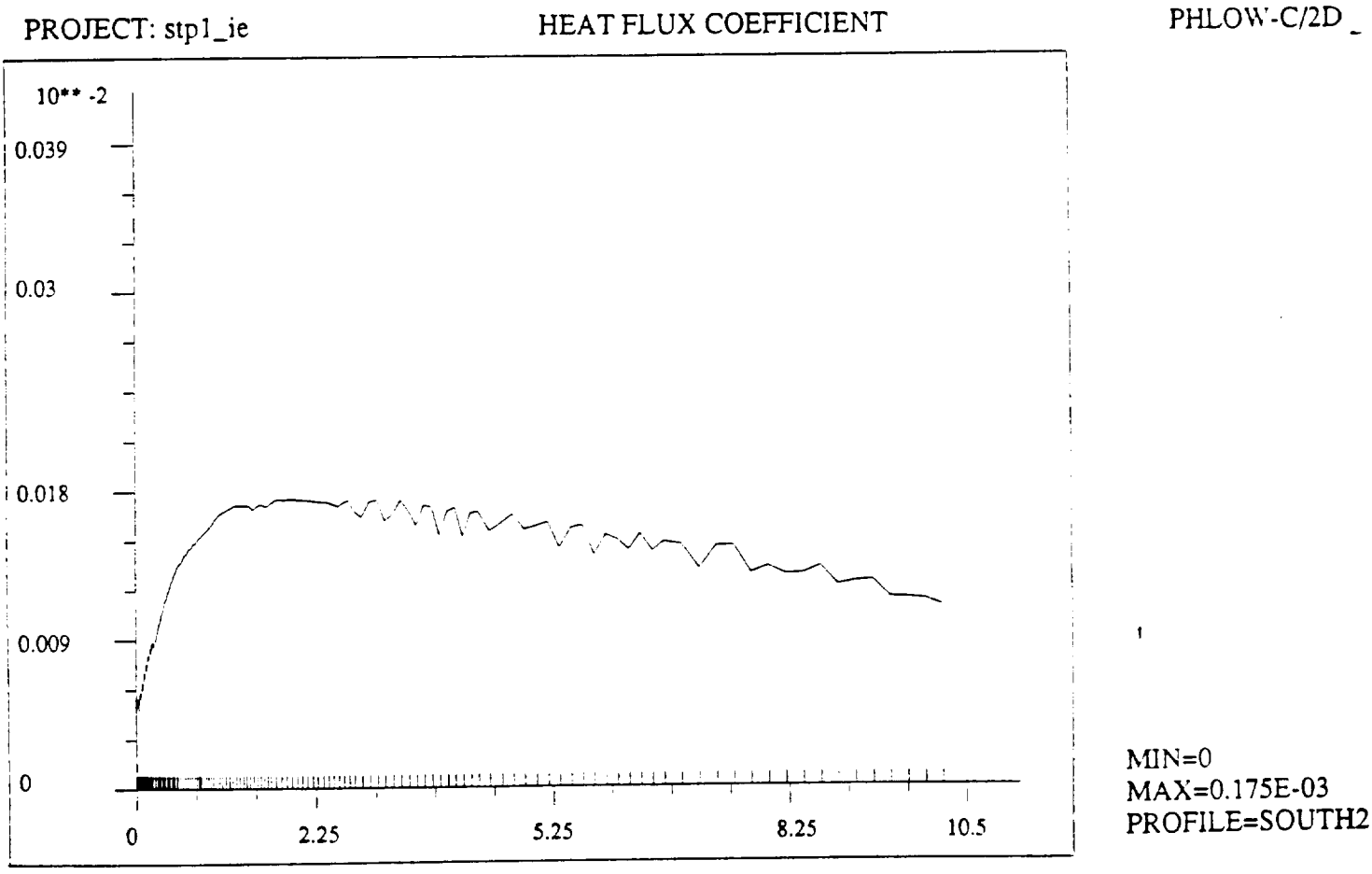


Figure 7.96: Heat flux coefficient profile along the wall downstream of the backstep, M3-mesh.

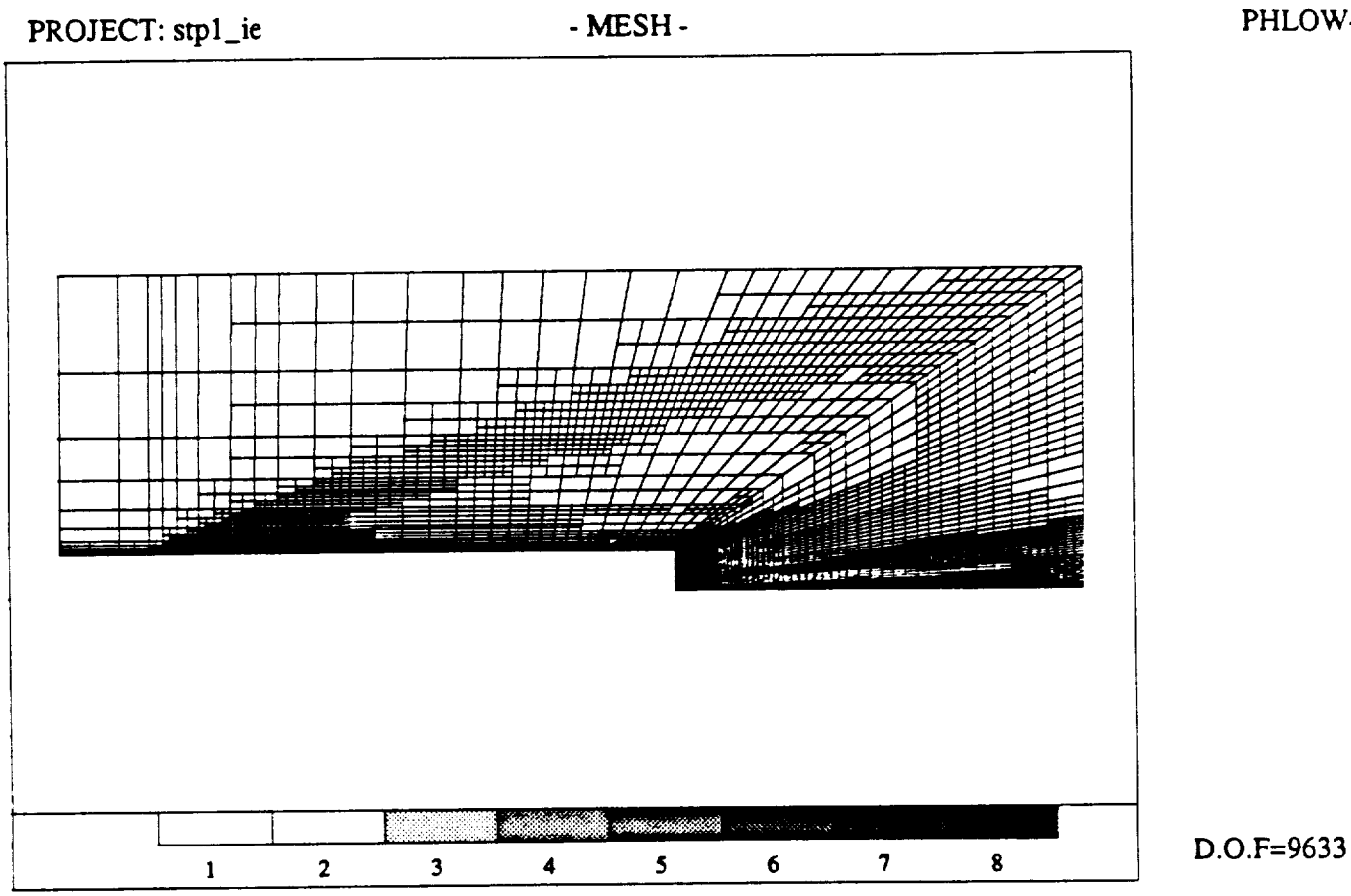


Figure 7.97: The final h-adapted mesh (M4H-mesh).

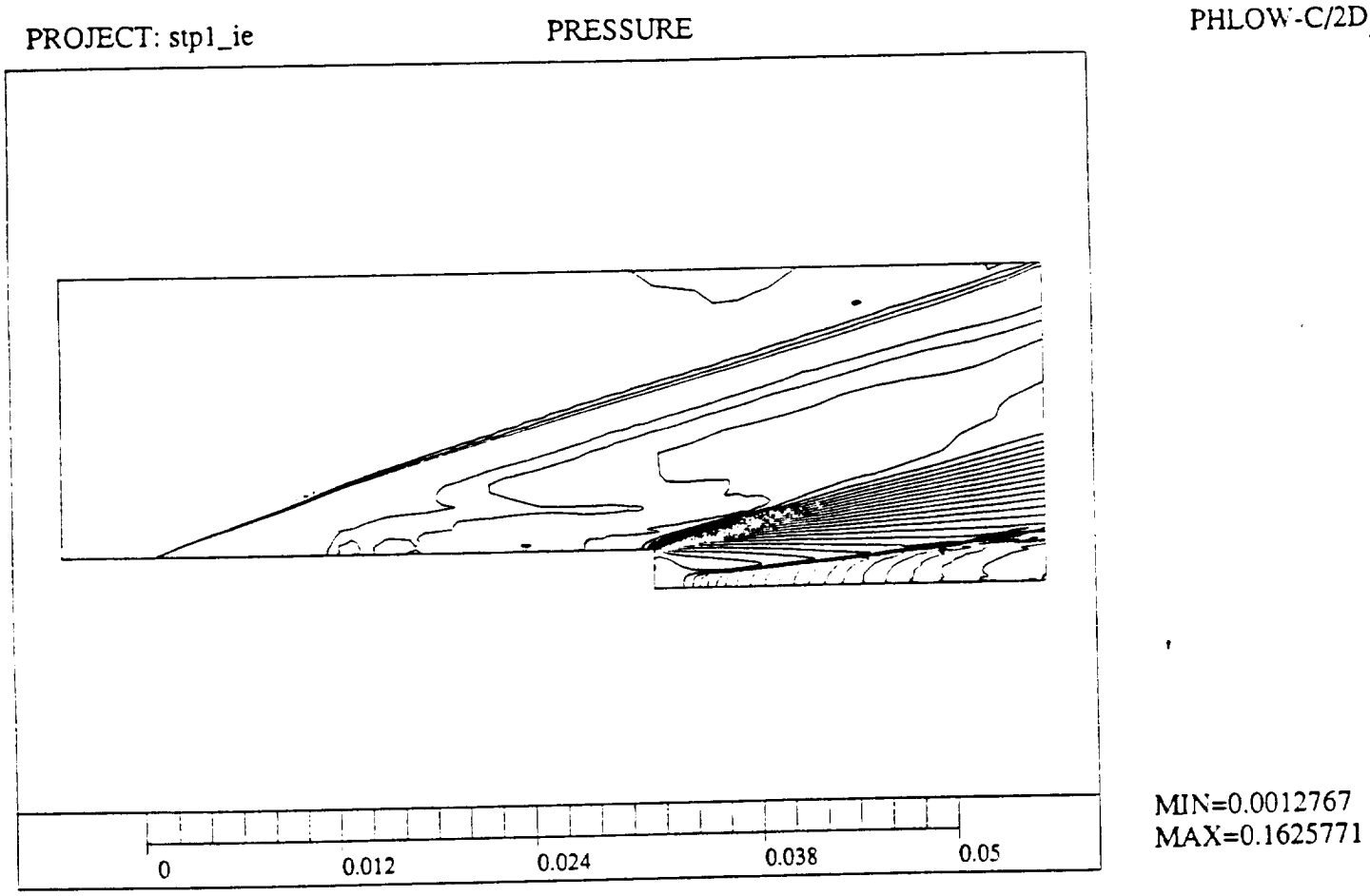


Figure 7.98: Pressure contours, M4H-mesh.

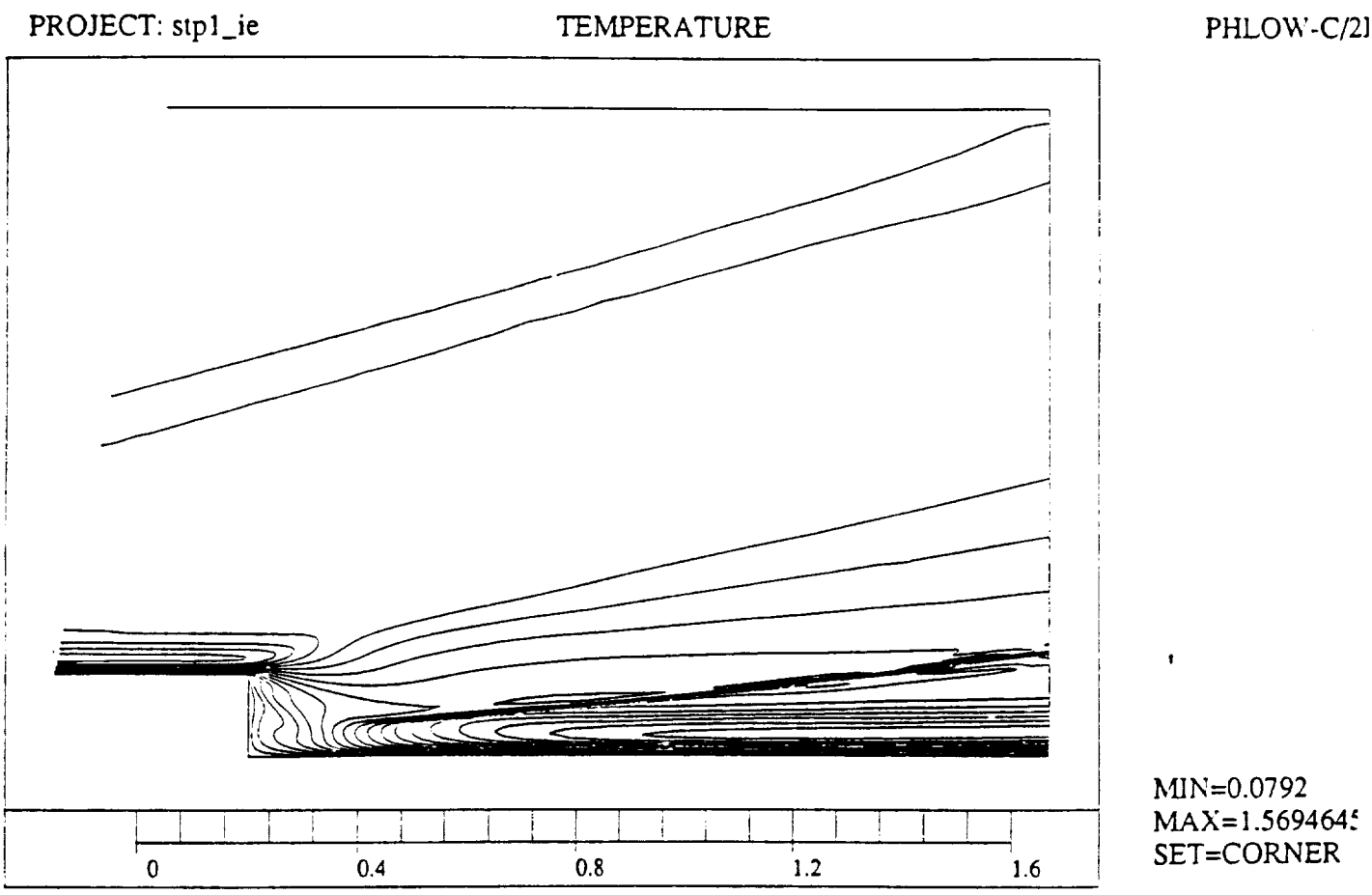


Figure 7.99: Temperature contours in the backstep region, M4H-mesh.

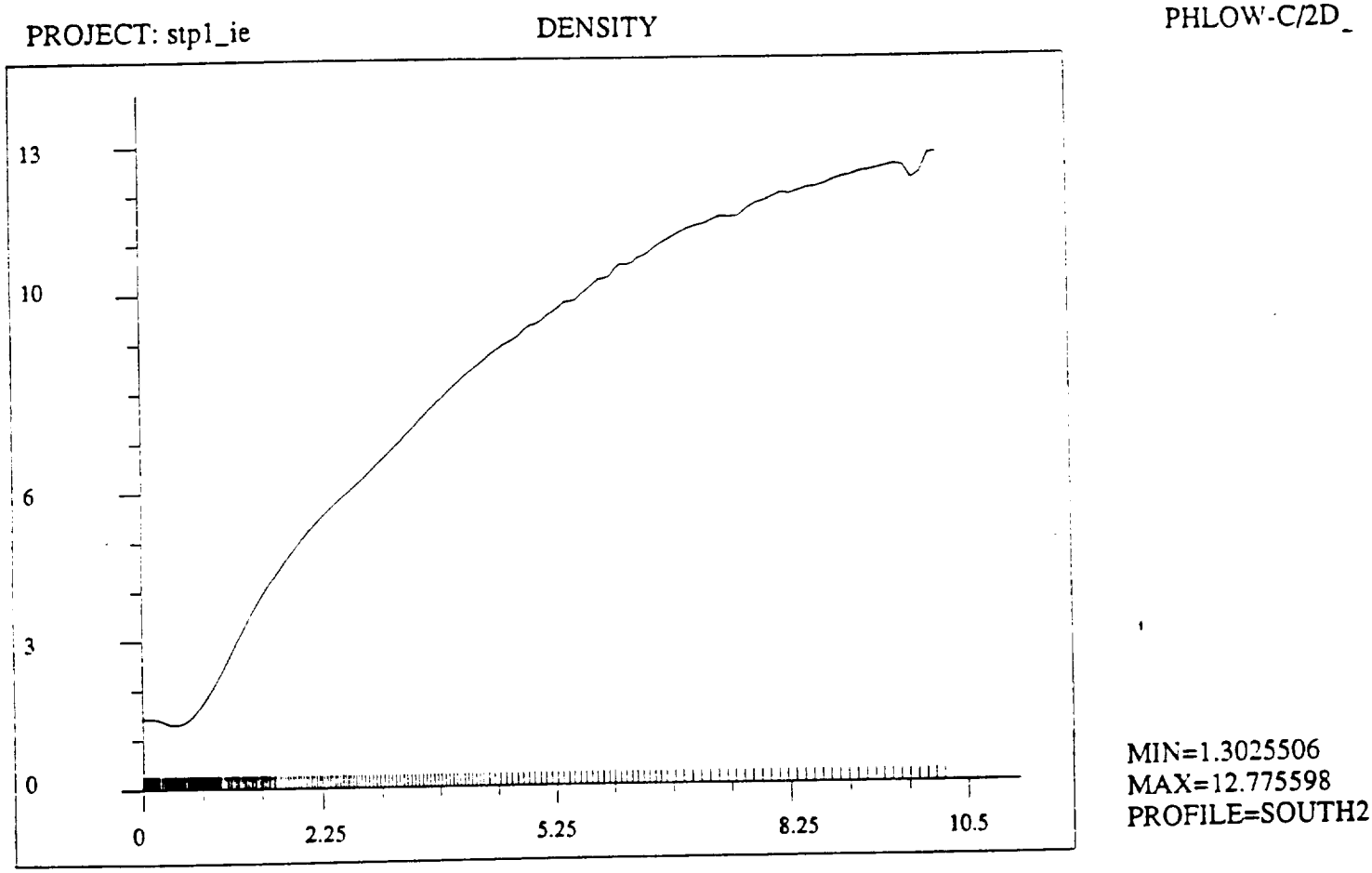


Figure 7.100: Density profile along the wall downstream of the backstep, M4H-mesh.

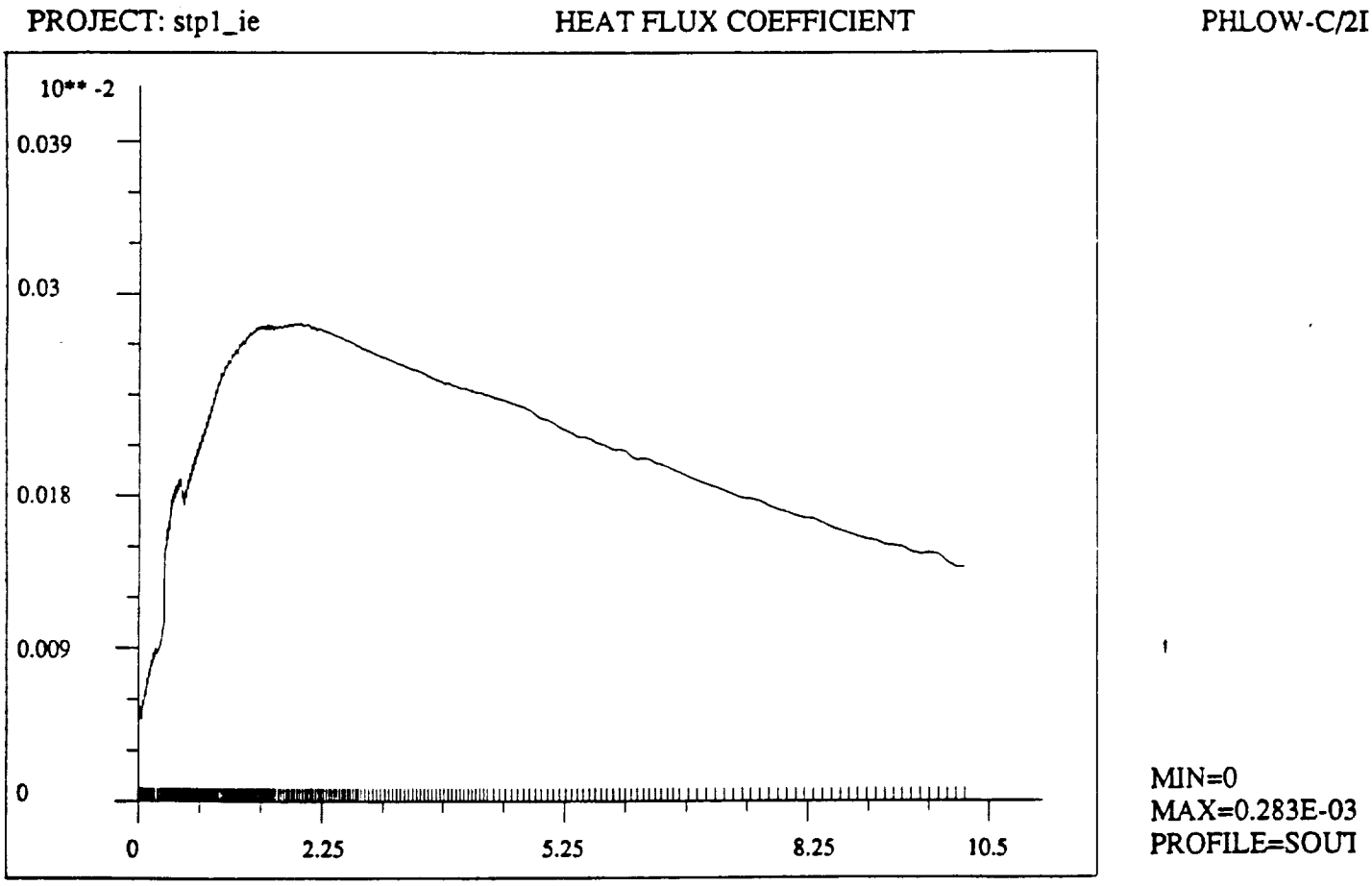


Figure 7.101: Heat flux coefficient profile along the wall downstream of the backstep, M4H-mesh.

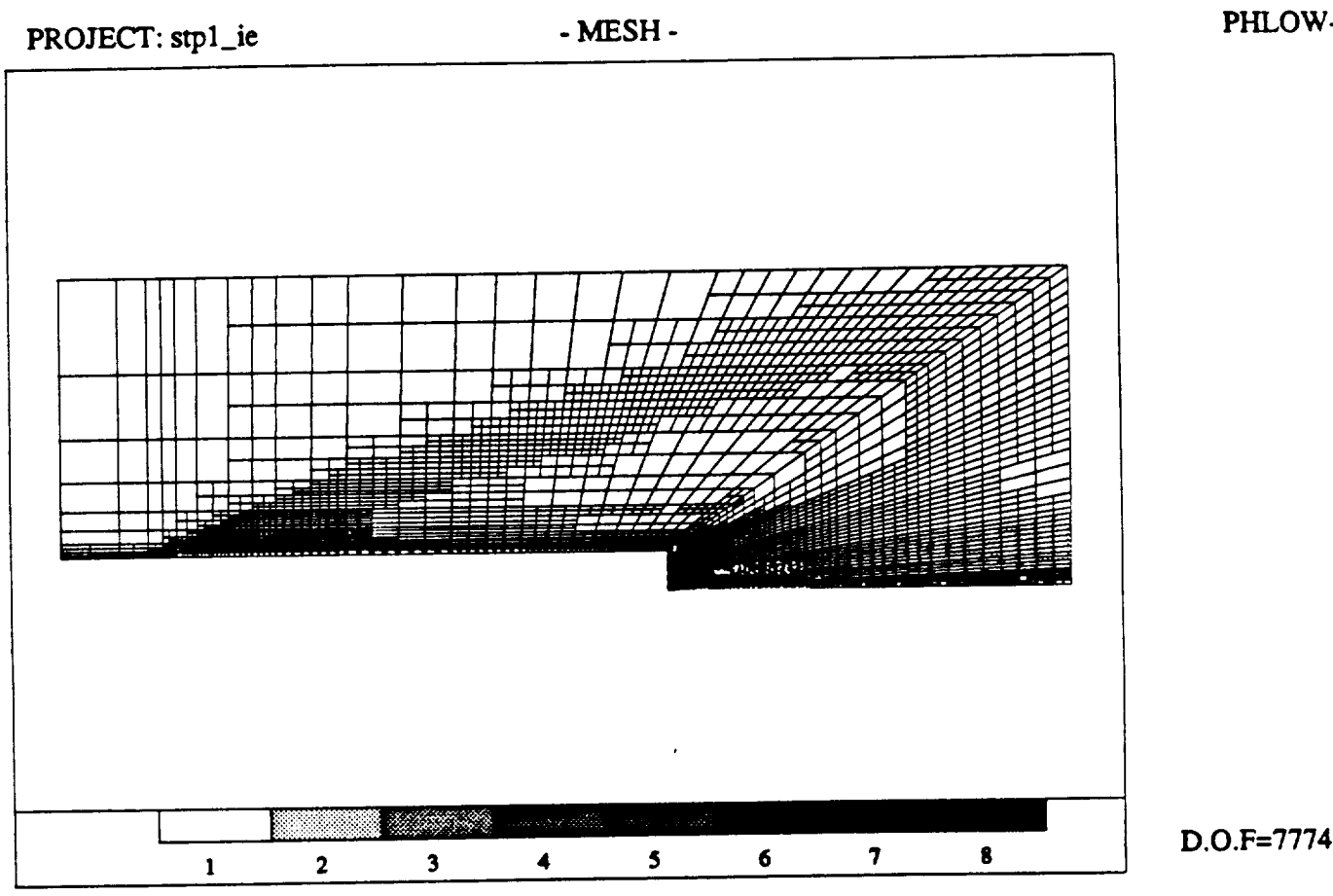


Figure 7.102: The final p-adapted mesh (M4P-mesh).

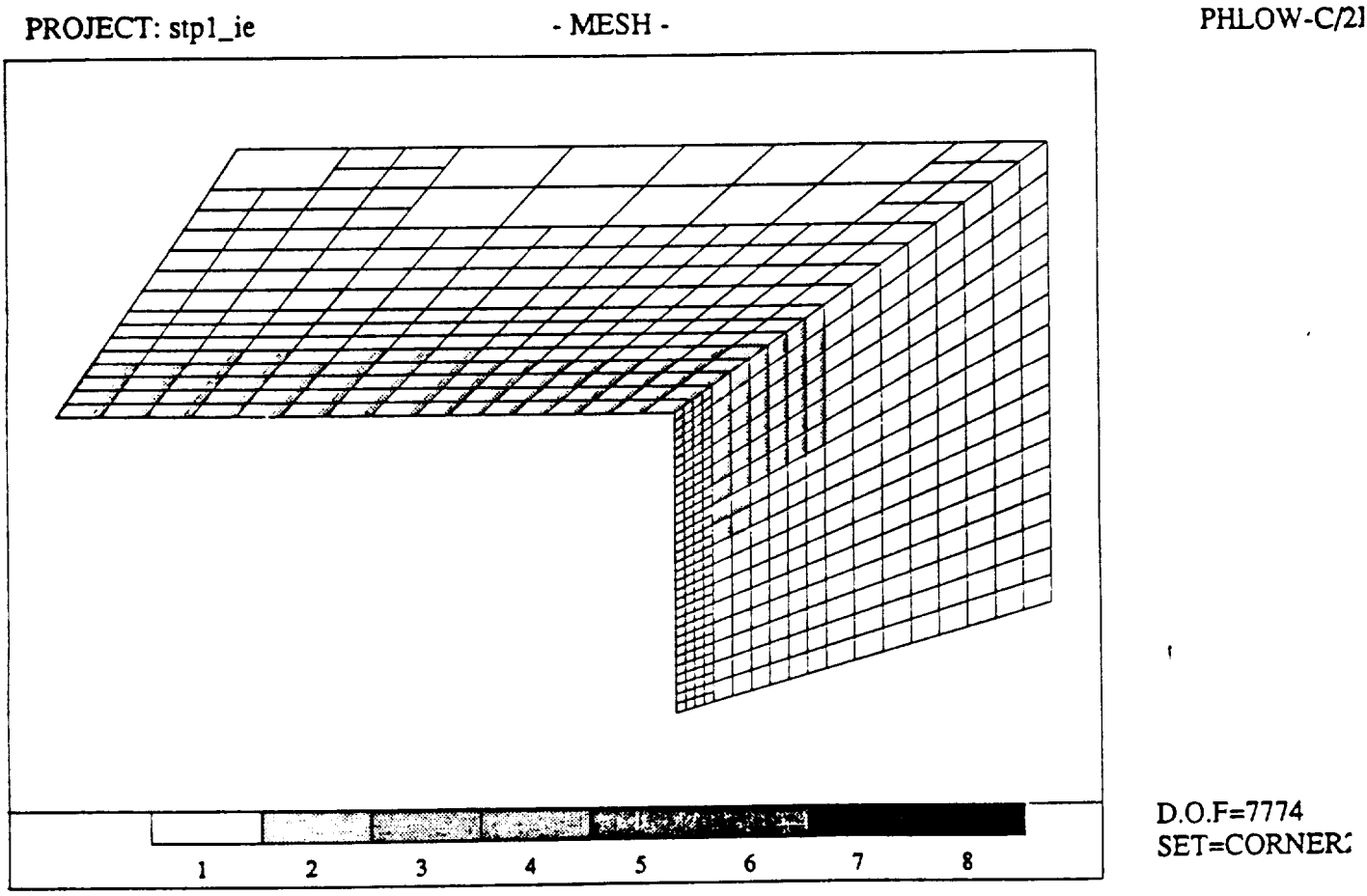


Figure 7.103: Blowup of M4P-mesh around the top corner of the backstep.

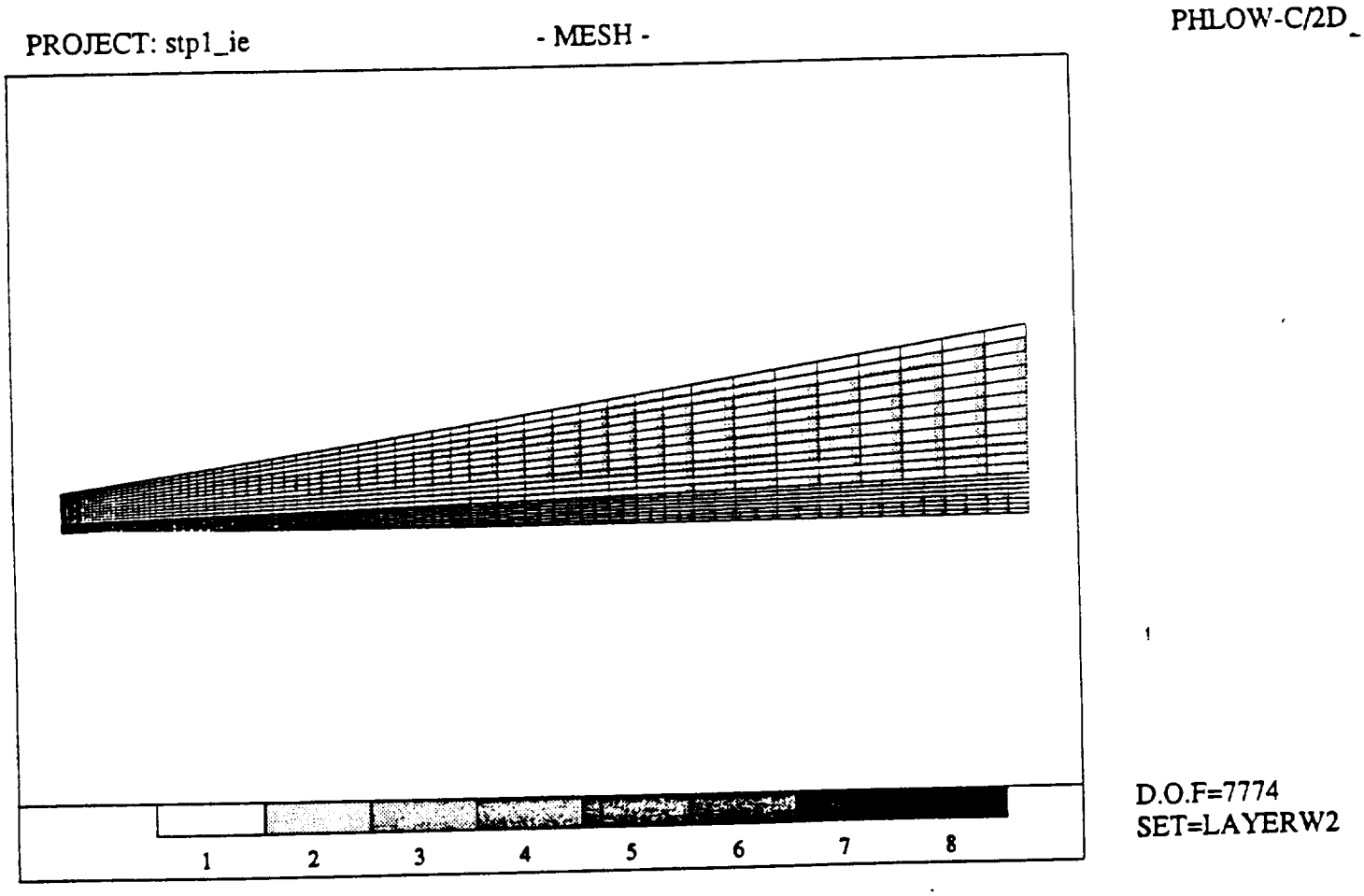


Figure 7.104: Blowup of M4P-mesh near the wall downstream of backstep.

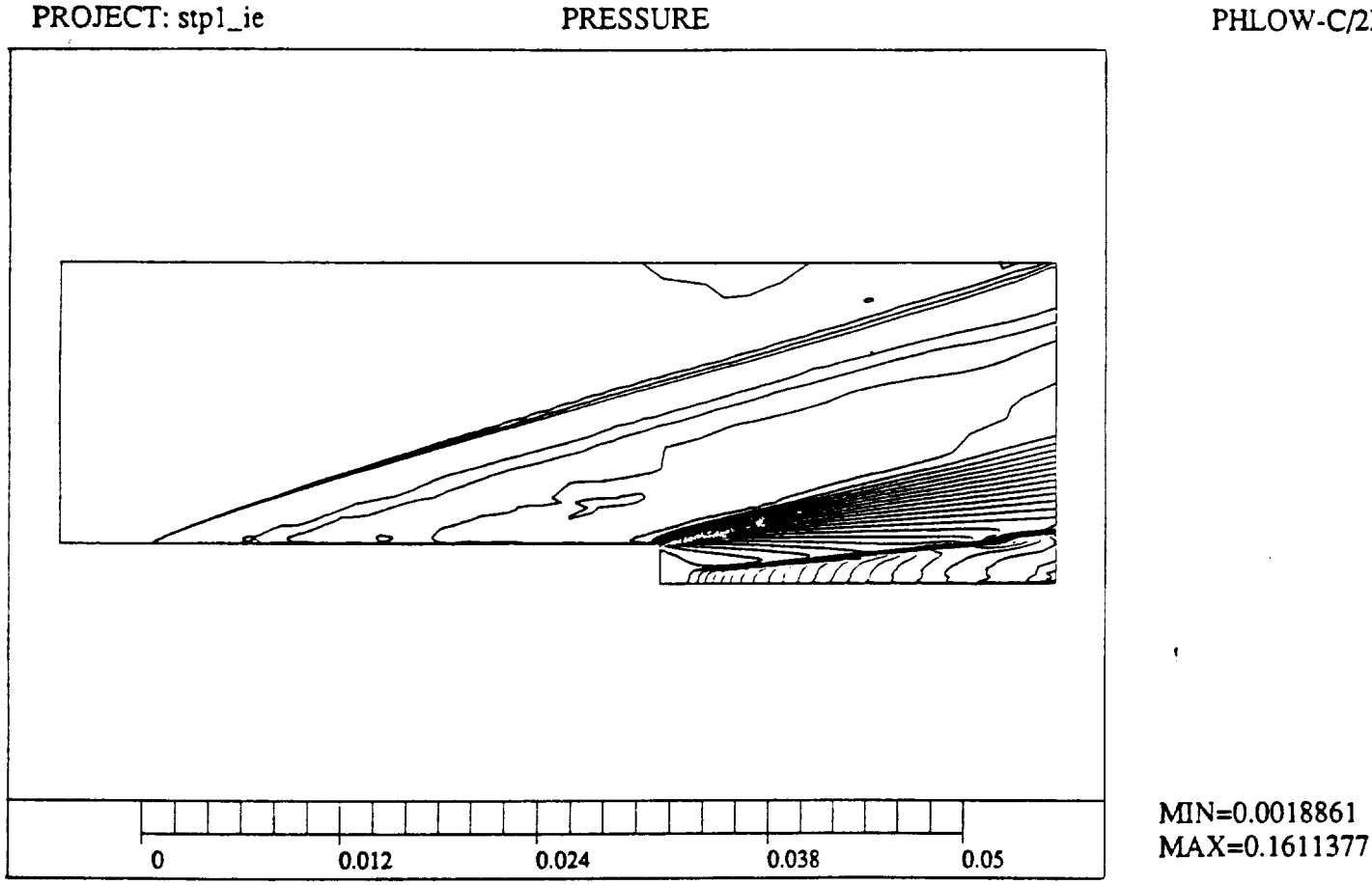


Figure 7.105: Pressure contours, M4P-mesh.

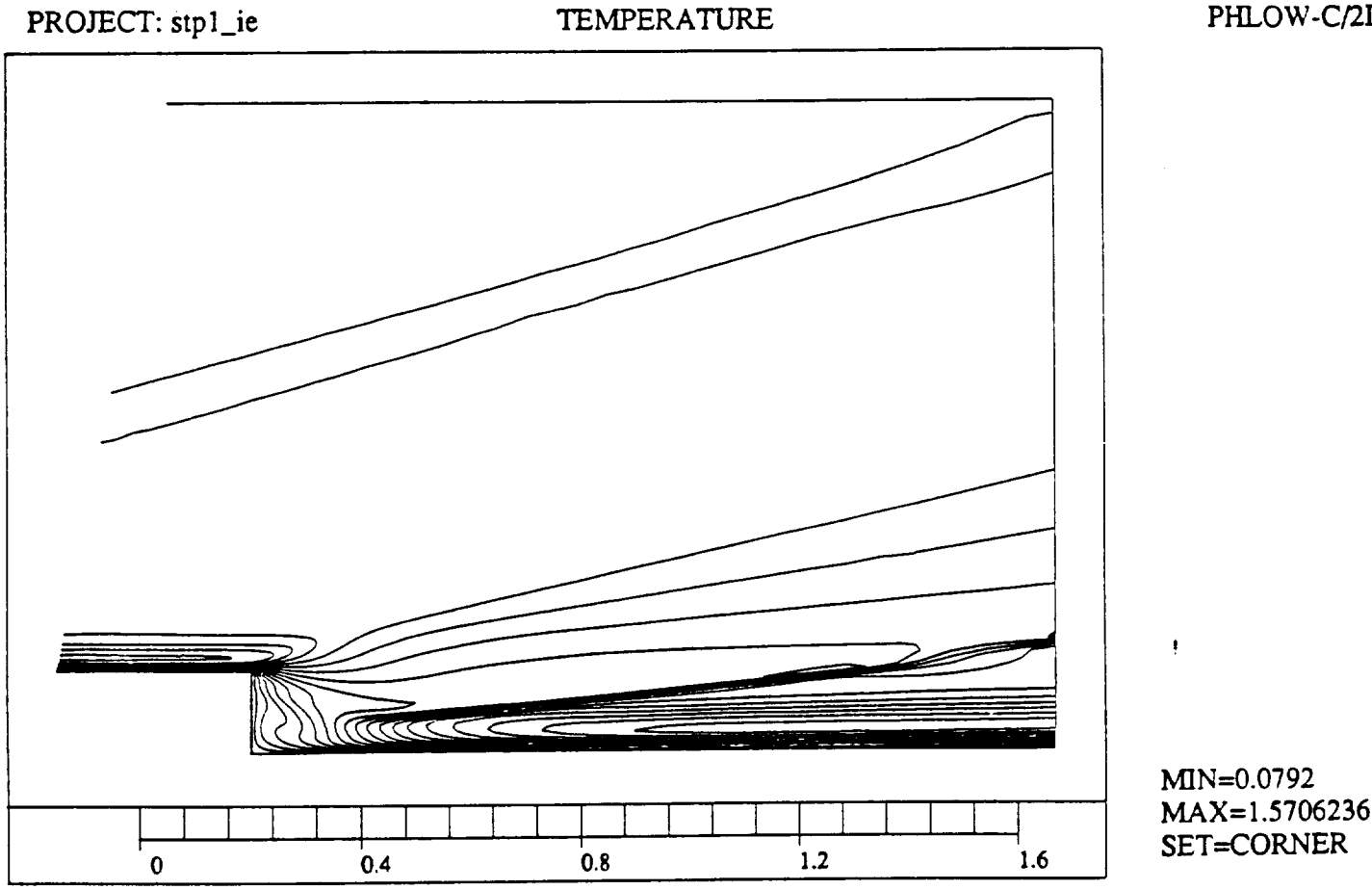


Figure 7.106: Temperature contours in the backstep region, M4P-mesh.

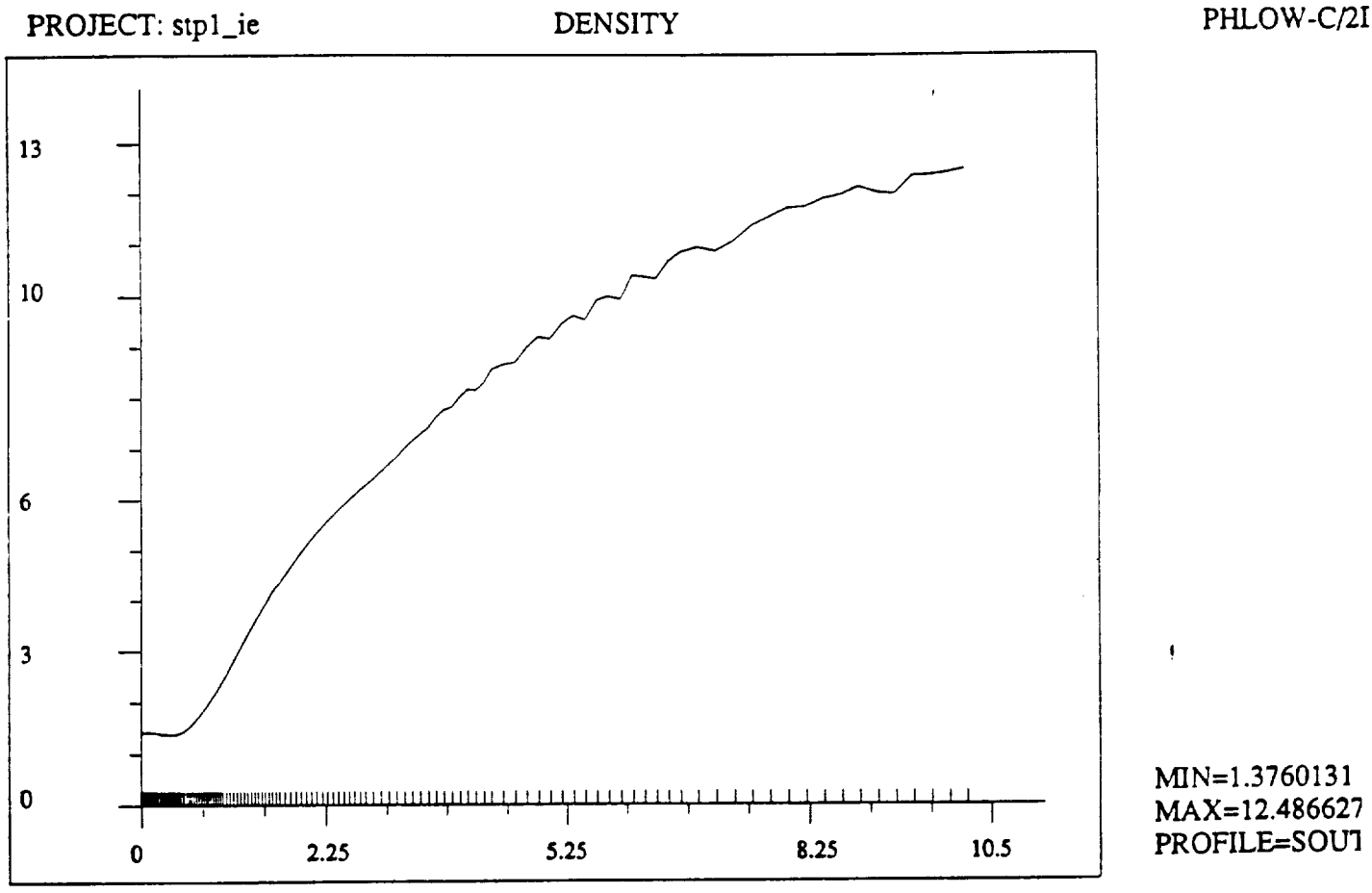


Figure 7.107: Density profile along the wall downstream of the backstep, M4P-mesh.

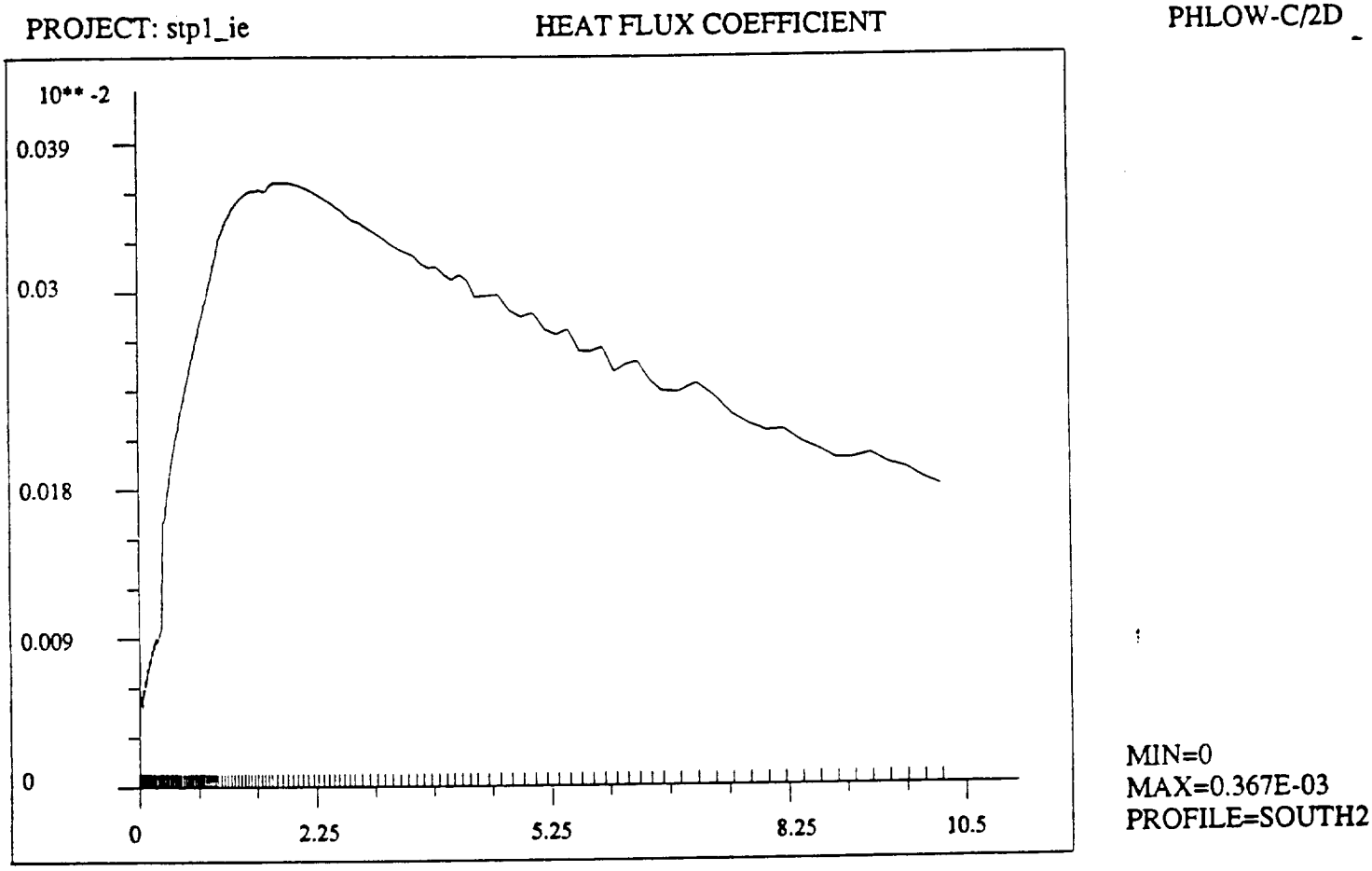


Figure 7.108: Heat flux coefficient profile along the wall downstream of the backstep, M4P-mesh.

Example 12: Double Swept Wedge Corner Flow Problem, $M = 3$

During the last year of the project, we have also initiated the solution of a three-dimensional benchmark problem, which involves modeling of the invicid flow past a wedge consisting of two planes inclined at different angles to the direction of the flow. The geometry of the problem is rather simple and is shown in Fig. 7.109. Due to plannar symmetry of this geometry the problem was solved only in one half of the computational domain with appropriate symmetry boundary conditions.

Up to date, we have only performed an initial solution of this problem on a rather coarse mesh. The corresponding solution, contours of density, and an h-adaptive mesh of linear elements are presented in Fig. 7.110. The flow, as expected is characterized by a skew shock which leaves the computational domain without any reflexions. Obviously, the results are far from converged on such a coarse mesh, however, the general shock structure and other features of the flow appear to be developing correctly. Further computations for this and other three-dimensional problems will be performed in the next year of the project, after completion of development of efficient three-dimensional implicit/explicit algorithms.

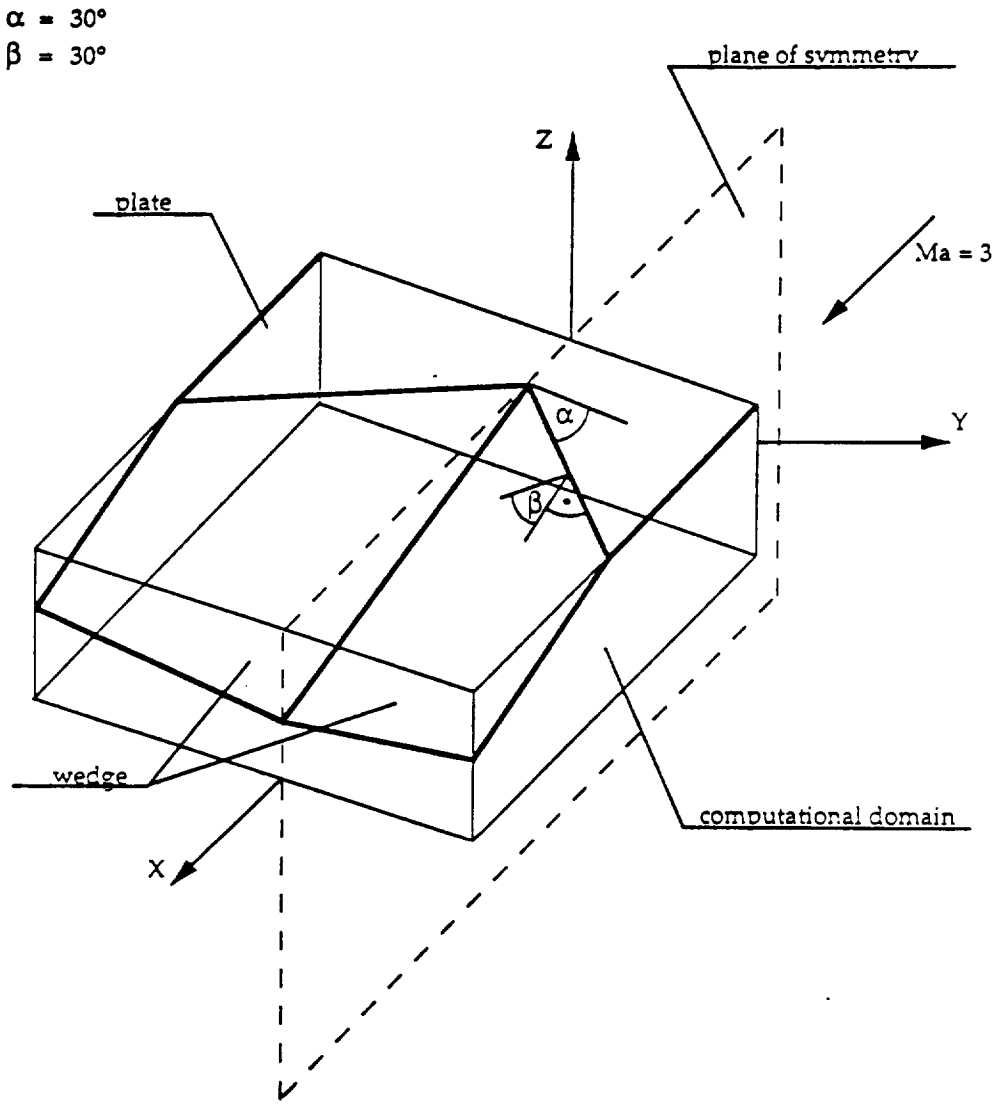


Figure 7.109: Double swept wedge, geometry and far-field conditions.

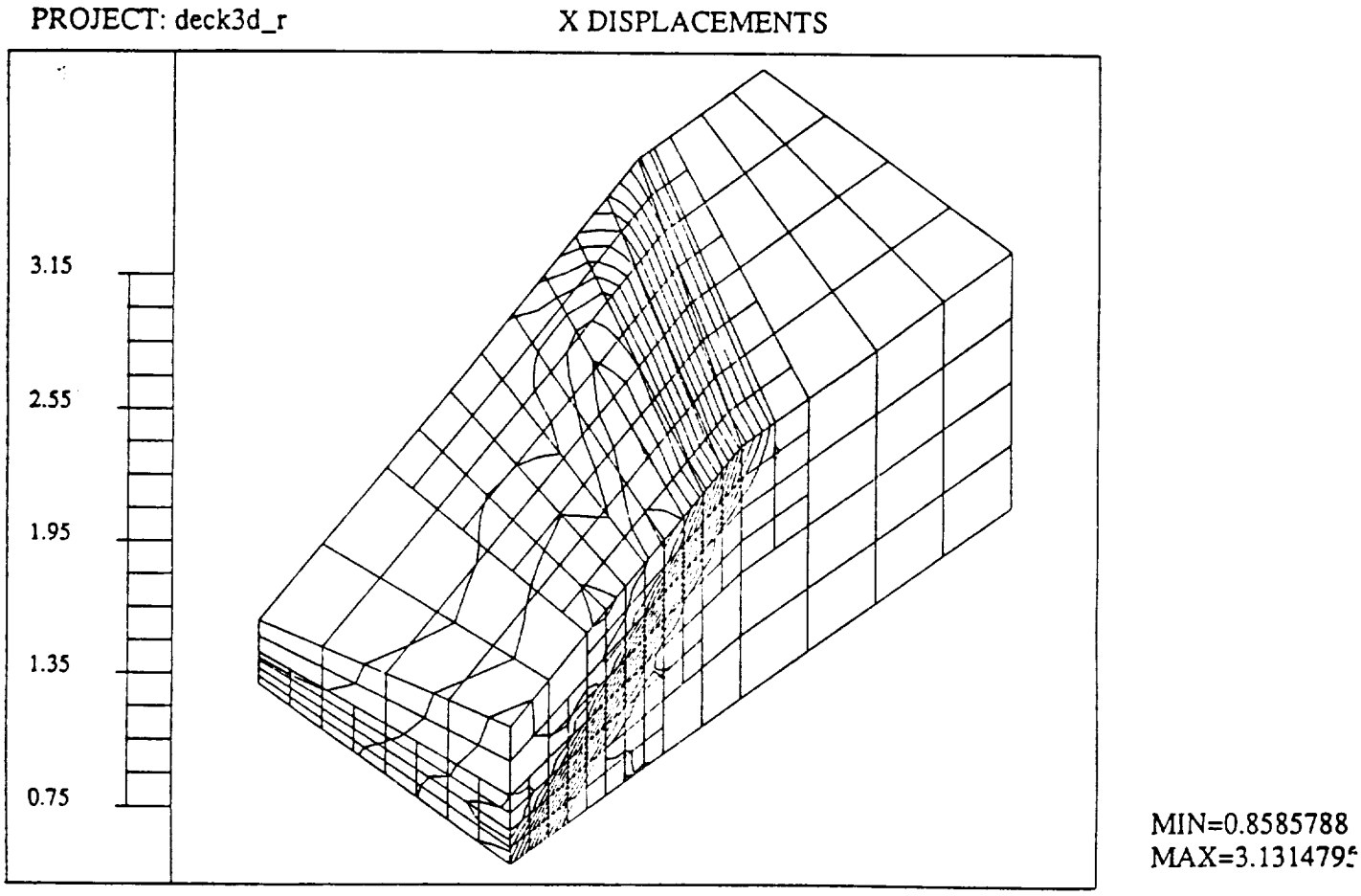


Figure 7.110: Double swept wedge, $M = 3$, density contours and h-adapted mesh of linear elements.

8 Phase II Project Summary and Future Directions

The computational results obtained for the various test cases and the theoretical advances made in the area of adaptive finite element methods over the past four years has been quite encouraging. They indicate that the *h-p* finite element method is not only a feasible approach for solving hypersonic flows but when combined with an implicit/explicit solution methodology provides an optimal framework for systematically changing the structure of the computational mesh to provide highly accurate numerical results with a minimal number of degrees of freedom and at minimum computational cost.

The technical efforts over the past year have focused on two topics which are specifically related to the overall performance of the flow solver. The first of these issues is that of directionally-dependent error estimation schemes which in general focused on an automated procedure for choosing an appropriate direction for *p*-enrichment. The algorithm selected herein uses a residual error estimate in conjunction with gradients of the residual error estimator or gradients of the solution (either of which may be selected by the user.) The residual error estimate itself is used to identify elements of the computational domain with relatively high errors. Among the group of elements with high errors a directional pointer, based on the gradients of a specified quantity, is obtained which indicated an optimal direction (or directions) in the master element for *p*-enrichment. Several numerical results have shown that the algorithm in general selects directions for enrichment that are normal to boundary layers and/or normal to shocks. In regions where point types of singularities exist, the indicator tends to select isotropic types of refinement.

The second major topic addressed over the past year, which is related to the overall performance of the flow solver, is that of implicit/explicit solution procedures. The general idea behind this methodology is that there is often a large diversity of stable time steps for various parts of the computational mesh. If one uses an implicit method in regions of the mesh with relatively severe time step restrictions and an explicit solution method in other regions of the mesh where the stability restrictions are not as severs then an optimal balance of computational effort is achieved within a single time step. Such an algorithm based on a general class of implicit Taylor-Galerkin algorithms was implemented within the context of the two-dimensional *h-p* flow solver. Based on the results of several test problems which employed this algorithm, an average computational savings of about 25 percent was achieved when compared with fully explicit algorithm and about 60 percent when compared with fully implicit algorithm used previously. Note that higher computational savings were obtained on problems with larger variations in the mesh size (see Section 7 for specific details).

The results obtained over the past year for the test problems and the benchmark problems in general indicate that the implicit/explicit methodology is a key component of an efficient

solution process. It provides a second level of optimization whereby not only an optimal mesh is used in the solution sequence but also an optimal time stepping procedure is employed. The proposed next phase of the effort will focus on extending the implicit/explicit solver methodology and the directional error estimation strategies to the three dimensional case. Based on our results over the past year the effort should be highly successful and may even provide more computational savings than in the two dimensional case.

9 References

1. Anderson, D. A., Tannehill, J. C., and Fletcher, R. H., **Computational Fluid Mechanics and Heat Transfer**, McGraw-Hill, N.Y., 1984.

2. Babuška, I., and Guo, B., "The *h-p* Version of the Finite Element Method for Problems With Nonhomogeneous Essential Boundary Conditions," *Comp. Meth. in Appl. Mech. and Engrg.*, Vol. 74, pp. 1–28, 1989.

3. Babuška, I., and Suri, M., "The *h-p* Version of the Finite Element Method With Quasiuniform Meshes," *Mathematical Modeling and Numerical Analysis*, **21** (2), pp. 199–238, 1987.

4. Bank, R. E., Sherman, A., H., and Weiser, A., "Refinement Algorithms and Data Structures for Regular Mesh Refinement," in **Scientific Computing**, R. Stepleman, *et al.* (Eds), IMACS, North Holland, 1983.

5. Carter, J. E., "Numerical Solutions of the Navier-Stokes Equations for the Supersonic Laminar Flow Over a Two–Dimensional Compression Corner," NASA Technical Report R-385, July 1972.

6. Courant, R., Friedrichs, K. O., and Lewy, H., Translated to: "On the Partial Difference Equations of Mathematical Physics," *FBM, J. Res. Dev.*, Vol. 11, pp. 215–234, 1967.

7. Demkowicz, L., and Oden, J. T., "A Review of Local Mesh Refinement Techniques and Corresponding Data Structures in *h*-type Adaptive Finite Element Methods," TICOM Report 88-02, The Texas Institute for Computational Mechanics, The University of Texas at Austin, Texas, 78712.

8. Demkowicz, L., Oden, J. T., and Rachowicz, W., "A New Finite Element Method for Solving Compressible Navier-Stokes Equations Based on an Operator Splitting Method and *h-p* Adaptivity," *Comp. Meth. in Appl. Mech. and Engrg.* Vol. 84, 1990, pp. 275–326.

9. Demkowicz, L., Oden, J. T., Rachowicz, W., and Hardy, O., "Toward a Universal *h-p* Adaptive Finite Element Strategy. Part 1: Constrained Approximation and Data Structure," *Comp. Meth. in Appl. Mech. and Engrg.*, **77**, pp. 79–112, 1989.

10. Demkowicz, L., Oden, J. T., Rachowicz, W., and Hardy, O., "An *h-p* Taylor-Galerkin Finite Element Method for Compressible Euler Equations" *Comp. Meth. in Appl. Mech. and Engrg.* Vol. 88, **3** 1991, pp. 363–396.

11. Devloo, Ph., Oden, J. T., and Pattani P., "An *h-p* Adaptive Finite Element Method for the Numerical Simulation of Compressible Flow," *Comp. Meth. in Appl. Mech. and Engrg.*, Vol. 70, pp. 203–235, 1988.

12. Dutt, P., "Stable Boundary Conditions and Difference Schemes for Navier-Stokes Equations," *SIAM J. Numer. Anal.*, Vol. 25, No. 2, 1988, pp. 245–267.

13. Gustafsson, B., and A. Sudström, "Incompletely Parabolic Problems in Fluid Dynamics," *SIAM J. Appl. Math.*, Vol. 35, No. 2, September 1978, pp. 343–357.

14. Harten, A., "On the Symmetric Form of Systems of Conservation Laws With Entropy," *Journal of Computational Physics*, Vol. 49, **4**, pp. 151–164, 1983.

15. Hassan, O., Morgan, K., and Peraire, J., "An Implicit Finite Element Method for High Speed Flows," AIAA Paper 90-0402, AIAA 28th Aerospace Sciences Meeting, Reno, Nevada, 1990.

16. Hughes, T. J. R., Franca, L. P., and Mallet, M., "New Finite Element Formulation for Computational Fluid Dynamics: I. Symmetric Forms of the Compressible Euler and Navier-Stokes Equations and the Second Law of Thermodynamics," *Computer Methods in Applied Mechanics and Engineering*, **54**, pp. 223–234, 1986.

17. Jakubowski, A. K., and Lewis, C. H., "Experimental Study of Supersonic Laminar Base Flow with and without Suction," AIAA Journal, Vol. 11, No. 12, December 1973, pp. 1670–1677.

18. Klopfer, G. H., and Yee, H. C., "Viscous Hypersonic Shock–On–Shock Interaction on Blunt Cowl Lips," AIAA Paper 88-0233, 1988.

19. Kuruvila, G., and Anderson, J. D. Jr., "A Study of the Effect of Numerical Dissipation on the Calculation of Supersonic Seperated Flows," AIAA Paper No. 85-0301, 1985.

20. Lapidus, A., "A Detached Shock Calculation by Second Order Finite Difference," *J. Comp. Phys.*, Vol. 2, 1967.

21. Lax, P. D., and Wendroff, B., "Systems of Conservation Laws," *Comm. Pure Appl. Math.*, Vol. 13, pp. 217–227, 1960.

22. Lerat, A., "Implicit Methods of Second-Order Accuracy for the Euler Equations," *AIAA Journal*, 23, 1, pp. 33–40, 1985.

23. Löhner, R., Morgan, K., and Peraire, T., "A Simple Extension to Multidimensional Problems of the Artificial Viscosity Due to Lapidus," *Comp. in Appl. Numer. Meth.*, Vol. 2, pp. 141–147, 1985.

24. Oden, J. T., Demkowicz, L., Rachowicz, W., and Westermann, T. A., "*A Posteriori* Error Analysis in Finite Elements: The Element Residual Method for Symmetrizable Problems with Applications to Compressible Euler and Navier-Stokes Equations," *Comp. Meth. in Appl. Mech. and Engrg.*: Special Issue: **Reliability in Computational Mechanics**, Edited by J. T. Oden, Vol. 82, Nos. 1-3, pp. 183–204, 1990.

25. Oden, J. T., Demkowicz, L., Rachowicz, W., and Westermann, T. A., "Toward a Universal *h-p* Adaptive Finite Element Strategy. Part 2: *A Posteriori* Error Estimation," *Comp. Meth. in Appl. Mech. and Engrg.*, **77**, pp. 113–180, 1989.

26. Rheinboldt, W. C., and Mesztenyi, Ch. K., "On a Data Structure for Adaptive Finite Element Mesh Refinement," *ACM Transaction on Mathematical Software*, Vol. 6, No. 2, pp. 166–187, 1980.

27. Saad, Y., and Shultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comput.*, Vol. 7, No. 3, July 1986.

28. Strang, G., "On the Construction and Comparison of Difference Schemes," *SIAM J. Numer. Anal.*, Vol. 5, No. 3, pp. 506–517, September 1968.

29. Strikwerda, J. C., "Initial Boundary Value Problems for Incompletely Parabolic Systems," Ph.D. Thesis, Dept. of Math., Stanford University, Stanford, CA, 1976.

30. Thareja, R. R., Prabhu, R. K., Morgan, K., Peraire, J., and Soltani, S., "Applications of an Adaptive Unstructured Solution Algorithm to the Analysis of High Speed Flows," AIAA 90-0395, 1990.

31. Tworzydlo, W. W., Oden, J. T., and Thornton, E. A., "Adaptive Implicit/Explicit Finite Element Methods for Compressible Viscous Flows," *Comp. Meth. in Appl. Mech. and Engrg.*, Vol. 95, pp. 397-440, 1992.

32. Vemaganti, G., and Wieting, "Application of a Finite Element Algorithm for High Speed Viscous Flows Using Structured and Unstructured Meshes," AIAA 90-1648, 1990.

Appendix

A Performance Issues

This appendix contains a summary of the work performed at the Computational Mechanics Company, Inc. in the areas of iterative solution techniques for the *h-p* finite element method and static condensation for high order spectral elements. The goal of these studies was to improve the computational efficiency of the *h-p* finite element method through improved element conditioning and reduction of element storage requirements. Funding for this work was provided through NASA, the Department of Defense, and private funding, and was not a part of the research and development effort conducted as part of this project. The results, however, are related to this project and are included here for completeness.

A.1 Iterative Methods for the *h-p* Finite Element Method

This section presents some numerical studies on the performance of iterative solvers implemented in the *h-p* finite element environment. In particular, a study of the GMRES algorithm combined with various versions of the block Jacobi preconditioners was performed. This iterative solver is used to solve nonsymmetric systems of linear equations occurring in the finite element analysis of compressible and incompressible flows, as well as in certain problems in solid mechanics.

The most difficult issues in the iterative solution of these systems is related to the definition and implementation of the preconditioner. A good preconditioner must satisfy several somewhat contradictory requirements, in particular:

- It should improve the conditioning of the system to be solved.
- It must precondition out all the penalty terms used to enforce the boundary conditions and other constraints. This is because extremely large eigenvalues introduced by the penalty terms ruin the convergence of the GMRES accelerator.
- The submatrices inverted in the process of preconditioning should be relatively small to minimize the computational cost of preconditioning.

In order to reasonably compromise the above criteria, we have studied a few different preconditioners of the block Jacobi type. These preconditioners were based on the concept of a "patch," which is explained further in this section.

The following subsections discuss the basic algorithm, preconditioning, and numerical studies on the performance of the GMRES method.

A.1.1 Basic Description of the Iterative Method

GMRES is a method for solving non-symmetric linear systems of equations, as described in [23]. The following is a brief description of the method:

Consider the system of linear equations:

$$Ax = f$$

The algorithm of the truncated/restarted GMRES method consists of the following steps:

1. Start:

   Choose the initial guess x_0

   Compute the search direction:

   $$r_0 = f - Ax_0$$

   normalize:

   $$v_1 = \frac{r_0}{\|r_0\|}$$

2. Loop through the Gramm-Schmidt algorithm to calculate m Arnoldi vectors spanning the Krylov subspace.

   This is done by the iterative procedure:

   $$\begin{aligned} j &= 1, 2 \ldots m \\ h_{ij} &= (Av_j, v_i) \ ; \ i = 1, 2 \ldots j \\ \tilde{v}_{j+1} &= Av_j - \sum_{i=1}^{j} h_{ij} v_i \\ h_{j+1,j} &= \|\tilde{v}_{j+1}\| \\ v_{j+1} &= \frac{\tilde{v}_{j+1}}{h_{j+1,j}} \end{aligned}$$

3. Form the approximate solution

   $$x_m = x_0 + v_m y_m$$

   where y_m minimizes $\|\beta_e - \overline{H}_m y\|$, $y \in R^m$ (see [23] for details).

4. Restart:

   Compute $r_m = f - Ax_m$; if the residual is sufficiently small, then stop

   else compute:

$$x_0 = x_m$$

$$v_1 = \frac{r_m}{\|r_m\|}$$

   and go to 2.

A.1.2 Separating a Preconditioner From an Accelerator

The GMRES algorithm is used to solve a preconditioned system of equations:

$$D^{-1}(Kx - f) = 0$$

The idea of using a preconditioner is to replace the original system of equations $Kx - f = 0$ by the new one, as above, with the matrix $D^{-1}K$ characterized by a much lower condition number than that of K.

The preconditioner D can be constructed from any so–called "basic iterative method" as follows:

Given a basic iterative method for solving $Kx - f = 0$:

$$x_{n+1} = Gx_n + k$$

where G, k satisfy the consistency condition: $(G - I)x + k = 0$ is equivalent to $Kx = f$. We introduce D such that the above formula becomes:

$$x_{n+1} = D^{-1}[-(K - D)x_n + dk]$$

Therefore $G = -D^{-1}(K - D) \rightarrow D^{-1} = (I - G)K^{-1}$. In many situations, even when the basic iterative method does not converge, it can serve as a good preconditioner (such as the Jacobi algorithm applied to the mass matrix in three dimensions).

Solving the preconditioned system of equations $D^{-1}(Kx - f) = 0$ with an algorithm like GMRES requires performing the following operations:

$$y := D^{-1}Kx$$

$$r := D^{-1}(-Kx + f)$$

From the definition of D we find that instead of explicitly preconditioning the matrix K by D^{-1}, one can perform one step of the basic method to compute:

$$\begin{aligned} y &= x_n - x_{n+1} \quad \text{with } f = 0 \\ r &= x_n - x_{n+1} \quad \text{with the actual } f \end{aligned}$$

This means that the "basic iterative method" $x_{n+1} = Gx_n + k$ can be quite independent from the accelerator GMRES and the above operations can be performed just by calling from the GMRES a "basic iterative solver," block Jacobi or Gauss-Seidel in our case.

A.1.3 Patch Definition

The solver, in particular the block Jacobi preconditioner, uses a block of stiffness matrices which are associated with patches of elements. A patch is the assembly of nodes associated with a node being preconditioned. The block inverted in D^{-1} includes all the degrees of freedom nodes included in the patch. Three definitions of patches have been investigated:

- patches associated with global linear nodes,
- patches associated by single nodes, and
- a combination of (i) corner nodes on the boundary and (ii) interior nodes.

The following is a brief description of each of the patch definitions:

1. Patches associated with global linear nodes consist of all nodes connected with the central node through the edges of elements. An example of a patch 1 is presented in Fig. A.1, where:

   • = patch node
   x = activated node

2. Patches associated with single nodes (Fig. A.2). An active node is the patch node.

3. Version 3 combines patch 1 for boundary nodes and patch 2 for interior nodes. An example of this type of patch is presented in Fig. A.3 where • = patch node and x = activated node.

Patches of the first kind provide a quite powerful preconditioning. Its major drawback is the large size of the patch stiffness matrix, especially for high p and three-dimensional elements.

• = patch node
x = activated node

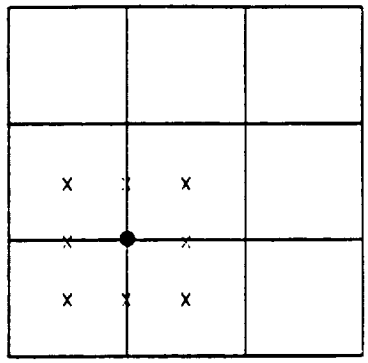


Figure A.1: Patch type 1.

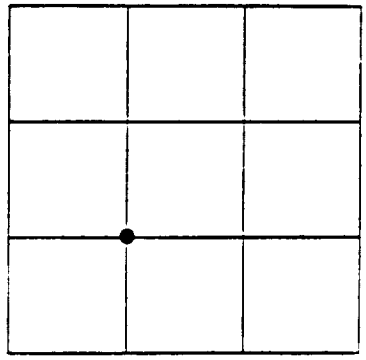


Figure A.2: Patch type 2.

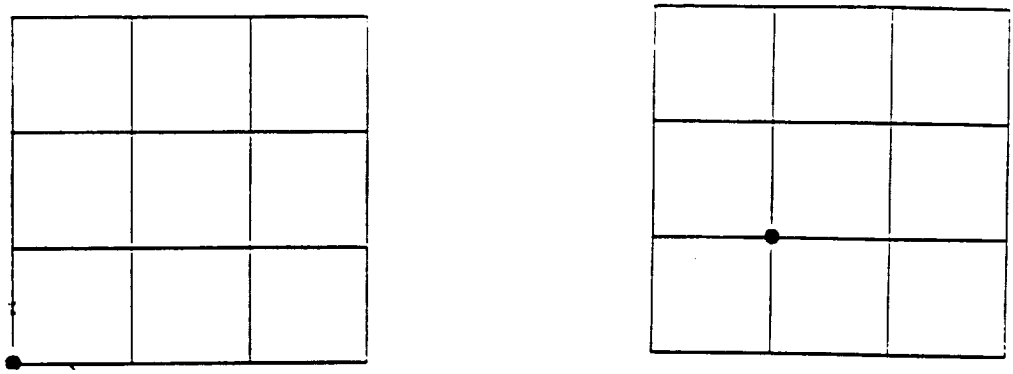


Figure A.3: Patch type 3. (a) Boundary Node. (b) Interior Node.

Patches of the second kind introduce small stiffness matrices. It is, however, not very effective, as it does not precondition out the penalty terms used to enforce boundary conditions for higher orders of approximation.

Patches of the third kind combine smaller patches on the interior and larger patches on the corner nodes of the boundary. It does precondition against the penalty terms on the boundary, however, it is not as effective as patches of the first kind. Note that for linear elements, patches of the first, second, and third kind all produce the same results.

A.1.4 Numerical Results

In this section, some sample results are presented from our study on the performance of iterative methods (i.e., GMRES accelerator and block Jacobi preconditioner). This performance study consists of a convergence check and a timing study where a comparison with a direct frontal solver is made and also the effects of using different kinds of patches is studied. The test problems considered range in complexity from a simple L_2 projection to problems in fluid mechanics (both incompressible and compressible).

Example 1. L_2 Projection

This problem is a two–equation L_2 projection case: $x^2 + y^2 = 1.0$ on a square domain. It is the simplest of the test cases and is used to verify the basic performance of the iterative solver. The mesh consists of second order polynomial element shape functions. There are 4 elements and 25 degrees of freedom. An example of the mesh is shown in Fig. A.4.

From Table A.1 it can be seen that only the iterative solver using patches of the first kind converges. It takes 2.5 times longer to converge as compared to the frontal solver. This is not a surprising result, however, because for small patches, direct solvers usually perform better than iterative solvers.

Example 2. Incompressible Fluid Mechanics

Here we present the timing results for a two–dimensional driven cavity flow. The geometry and boundary conditions of the problem are simple and are very well known. For the case under consideration, the flow field moves in the positive x-direction and the Reynolds number is 1.0. The mesh consists of second order polynomial element shape functions for the velocity field and first order polynomial element shape functions for the pressure field. There are 64 elements and 289 degrees of freedom per unknown. An example of the mesh is presented in Fig. A.5.

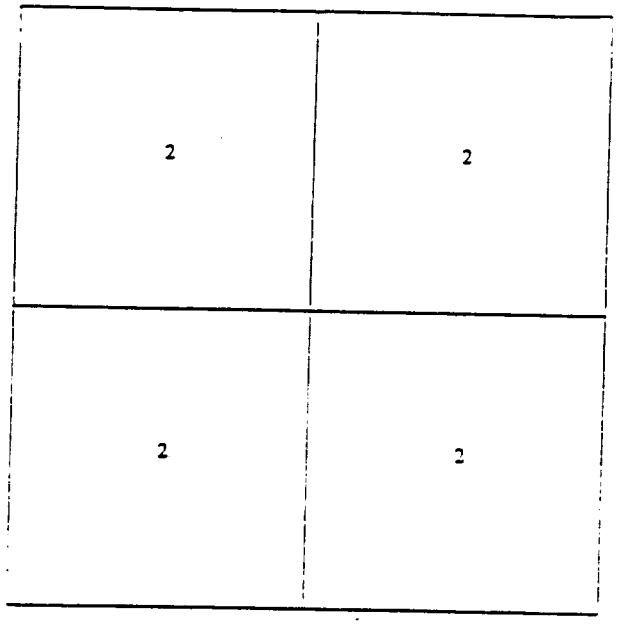


Figure A.4: Mesh for the L_2 projection.

SOLVER	TIME	ERROR MIN	ERROR MAX
FRONTAL	0.321597	0.0	2.0
GMRES P=1	0.797104	0.0	2.0
P=2	64.254	-0.03603	1.978011
P=3	2.8510	-0.03006	1.718309

Table A.1: Timing results for L_2-projection.

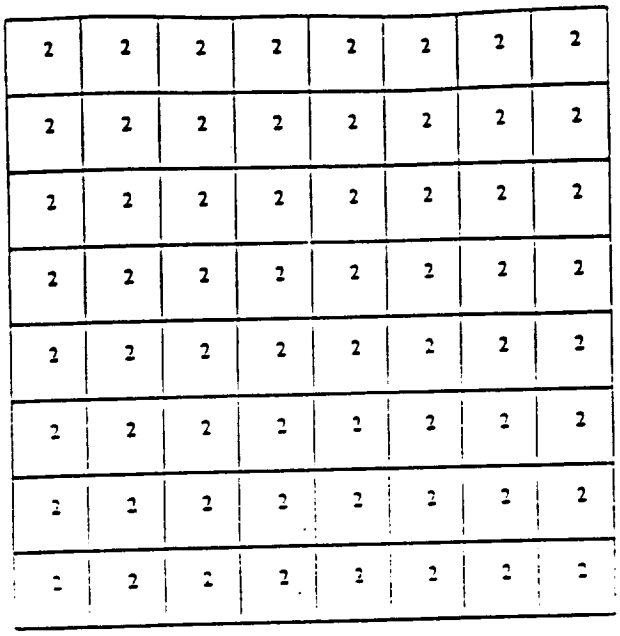


Figure A.5: Driven cavity mesh.

SOLVER	TIME	ERROR MIN	ERROR MAX
FRONTAL	33.52617	-0.347012	0.3440713
GMRES P=1	98.41675	-0.33872	0.3384091
P=2	250.2750	-0.08106	-0.08312
P=3	502.2729	-0.243403	0.243493

Table 2
Timing results for driven cavity problem.

From Table A.2 it can be seen again that only the iterative solver with patches of the first kind converges. It takes approximately 2.9 times longer to reach the same solution as the frontal solver. The iterative solver with patches of the second and third kind do not converge to the correct solution; however, the residuals of the GMRES does converge. As expected, the iterative solver with patches of the third kind converge closer to the solution as compared to patches of the second kind. The bad performance of preconditioners based on patch types 2 and 3 is probably caused by a complex coupling of pressures and velocities (through the gradient operator), which is not resolved by small patches for interior nodes.

Example 3. Compressible Fluid Mechanics

Presented here are the timing results for the standard wedge benchmark problem for inviscid supersonic flows. For this case, a 5 degree wedge with a uniform inflow and the Mach number of 5.0 was selected. The mesh consisted of 240 linear elements with 275 degrees of freedom per unknown. (The mesh is presented in Fig. A.6.) The problem was run for 300 time steps.

From Table A.3 it can be seen that there is no convergence check. This is due to the frontal solver and the iterative solver being converged to the same solution. As expected, the iterative solver converged irrespective of the type of patch used. The iterative solver using patches of the first kind converged the fastest followed by patches of the second kind, and lastly, patches of the third kind. The iterative solver converged 3.9 times faster than the frontal solver. The primary reason the iterative solver converged faster than the fronter solver is that the mass matrix for this case is symmetric and positive definite. A symmetric positive definite matrix has real eigenvalues and has a good conditioning number. These characteristics of the matrix bring about a faster convergence rates for iterative solvers.

Conclusions

The iterative solver (i.e., GMRES accelerator with block Jacobi preconditioner), utilizing patches of the first kind, converged to the same solution as the frontal solver for all cases under consideration. The iterative solver converges slowly if the matrix is ill–conditioned or if a penalty method applied to the boundary condition has not been preconditioned out. It was found that the iterative solver converges more rapidly than the frontal solver for well–conditioned matrices (the presence of mass matrix contributions on the left hand side is very beneficial). From the present study, the use of the iterative solver with patches of the second and third kind does not appear to be a good option.

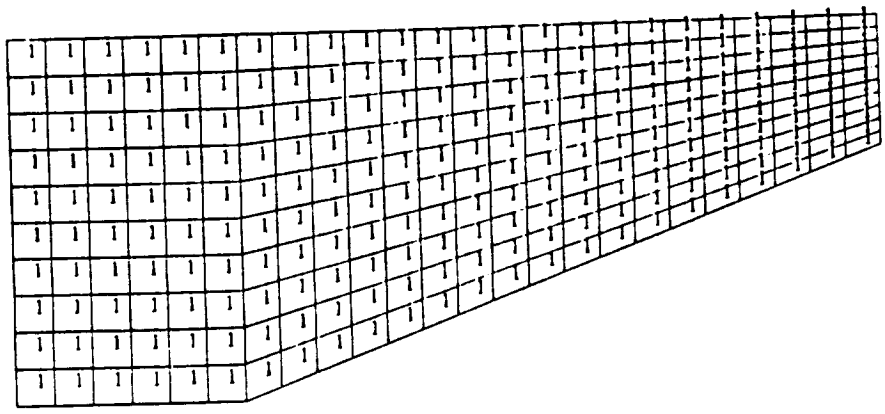


Figure A.6: 5 degree wedge mesh.

SOLVER	TIME
FRONTAL	6119.33
GMRES P=1	1576.29
P=2	1616.63
P=3	1721.78

Table A.3: Timing results for 5 degree wedge problem.

A.2 Matrix Condensation

One of the side effects of using higher order finite element approximations is the extremely large size of the element matrices that result, especially in three dimensions. For an eighth order element in three dimensions, this corresponds to a local vector with $729 \times \text{NDOF}$ where NDOF is the number of solution components. Converting this into storage requirements implies that approximately 0.5 megawords of memory are required for each component of the solution for a single element. To reduce this large memory requirement we are investigating the use of static condensation to eliminate the internal degrees of freedom before the element matrix is passed to the iterative solver. For the eighth order element mentioned above, this corresponds to the removal of $343 \times \text{NDOF}$ degrees of freedom at the element level which in turn reduces storage requirements of the element matrix to less than one third of the original size. In addition to reducing the element storage size, this technique also has the advantage of improving the conditioning of the global system by eliminating the coupling between nodal and central degrees of freedom and edge and central degrees of freedom.

Condensation Algorithm

Depending on which numerical algorithm is actually used in the solution of the governing equations, the resulting forms of the element local stiffness matrix and right hand sides may be written as

$$S_i = \begin{bmatrix} A_i & B_i \\ D_i & C_i \end{bmatrix} \qquad R_i = \begin{bmatrix} b_i \\ c_i \end{bmatrix}$$

where

- A_i are stiffness matrix components representing interactions among the internal degrees of freedom only.
- C_i are stiffness matrix components representing interactions among the side and nodal degrees of freedom.
- B_i and D_i are stiffness matrix components representing mixed types of interactions between internal and side and nodal degrees of freedom.
- b_i is the right hand side vector associated with the internal degrees of freedom.
- c_i is the right hand side vector correspnding to all other degrees of freedom not at the central node.

The process of condensation, or elimination of internal unknowns, entails computing the Schurr complement:

$$\tilde{C}_i = C_i - D_i A^{-1} B_i$$

and modifying the right hand side as follows

$$\tilde{c}_i = c_i - D_i A_i^{-1} b_i$$

These quantities are then normalized so that all local diagonal matrix entries are one.

$$\tilde{C}_i = E_i^{-\frac{1}{2}} \tilde{C}_i E^{-\frac{1}{2}} \qquad \tilde{c}_i = E^{-\frac{1}{2}} c_i \quad \text{where } E_i = \text{diag}\,(\tilde{c}_i)$$

Using these element matrices, a global system of the following form is obtained which no longer contains the internal degrees of freedom,

$$\tilde{S}\tilde{\alpha} = \tilde{r}$$

where

$$\tilde{S} = \sum_i \tilde{C}_i \quad \tilde{r} = \sum_i \tilde{c}_i$$

and $\tilde{\alpha}_i$ is a vector of unknowns associated with the side and nodal degrees of freedom.

The final step is the calculation of the internal unknowns α_i. These are computed using $\tilde{\alpha}_i$ by solving the local systems

$$A_i \alpha_i = b_i - B_i \tilde{\alpha}_i$$

Matrix Operations and Array Sizes

To solve a problem of q equations on a $n \times n$ mesh using uniform p order elements, the size of the global stiffness matrix is:

$$\left[\left((n+1)^2 + 2n(n+1)(p-1) + n^2(p-1)^2\right) \times q\right]$$

where $(n+1)^2$ represents the degrees of freedom associated with the combinations of the corners, $2n(n+1)(p-1)$ represents the degrees of freedom associated with the combinations of the edges, $n^2(p-1)^2$ represents the degrees of freedom associated with the center, and q is the number of equations.

For an 8×8 mesh with fourth order elements and two equations, one solves a system of 2178 equations. With condensation, this system is reduced to n^2 inversions of $[(p-1)^2 \times q]^2$ matrices and to solving a system of $[(n+1)^2 + 2n(n+1)(p-1)] \times q$ equations. For the sample case, this is 64 (18×18) matrices to invert which is equivalent to solving a system of 513 equations. Locally it reduces the stiffness matrices from $(p+1)^2 \times q$ to 4 pq, which is significant for high p. It is even more significant in three dimensions when it goes from $(p+1)^3 \times q$ to $(6p^2+2)q$.

Preliminary results using the static condensation procedure for L_2 projections has shown a slight increase in the converge rate which is most probably due to improved conditioning and up to 20 percent savings in computational time for larger p.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 21, 1993	Contractor Report

4. TITLE AND SUBTITLE
H-P Adaptive Methods for Finite Element Analysis of Aerothermal Loads in High-Speed Flows

5. FUNDING NUMBERS
C NAS1-18746

WU 506-40-21

6. AUTHOR(S)
H.J. Chang, J.M. Bass, W.W. Tworzydlo, and J.T. Oden

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Computational Mechanics Company, Inc.
7701 North Lamar, Suite 200
Austin, Texas 78752

8. PERFORMING ORGANIZATION REPORT NUMBER
TR-92-12

9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NASA
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING MONITORING AGENCY REPORT NUMBER
NASA CR-189739

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: George C. Olsen
Final Report

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unclassified-Unlimited

Subject Category 34

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The commitment to develop the National Aerospace Plane and Manuevering Reentry Vehicles has generated resurgent interest in the technology required to design structures for hypersonic flight. The principal objective of this research and development effort has been to formulate and implement a new class of computational methodologies for accurately predicting fine scale phenomena associated with this class of problems. The initial focus of this effort was to develop optimal h-refinement and p-enrichment adaptive finite element methods which utilize a-posteriori estimates of the local errors to drive the adaptive methodology. Over the past year this work has specifically focused on two issues which are related to overall performance of flow solver. These issues include the formulation and implementation (in two dimensions) of an implicit/explicit flow solver compatible with the hp-adaptive methodology, and the design and implementation of computational algorithms for automatically selecting optimal directions in which to enrich the mesh. These concepts and algorithms have been implemented in a two-dimensional finite element code and used to solve three hypersonic flow benchmark problems (Holden Mach 14.1, Edney shock on shock interaction Mach 8.03, and the viscous back-step Mach 4.08).

14. SUBJECT TERMS
hypersonic flows, h-p adaptive methods, directional p-enrichment, implicit/explicit methods

15. NUMBER OF PAGES
284

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		